LA-UR 95-1293

Compact Location Problems

Authors: S.O. Krumke, M.V. Marathe, H. Noltemeier,
V. Radhakrishnan, S.S. Ravi, D.J. Rosenkrantz

September 1996

LOS ALAMOS
NATIONAL LABORATORY

Compact Location Problems [5]

S. O. Krumke [1] M. V. Marathe [2] H. Noltemeier [1]
V. Radhakrishnan [3] S. S. Ravi [4] D. J. Rosenkrantz [4]

September 20, 1996

Abstract

We consider the problem of placing a specified number (p) of facilities on the nodes of a network so as to minimize some measure of the distances between facilities. This type of problem models a number of problems arising in facility location, statistical clustering, pattern recognition, and also processor allocation problems in multiprocessor systems. We consider the problem under three different objectives, namely minimizing the diameter, minimizing the average distance, and minimizing the variance. We observe that in general, the problem is NP-hard under any of the objectives. Further, even obtaining a constant factor approximation for any of the objectives is NP-hard.

We present a general framework for obtaining near-optimal solutions to the compact location problems for the above measures, when the distances satisfy the triangle inequality. We show that this framework can be extended to the case when there are node weights and also to the case when there is a distinguished set of nodes which must be chosen in the placement. We also investigate the complexity and approximability of the extension of these problems, when two distance values are specified for each pair of potential sites. In these cases, the goal is to a select a specified number of facilities to minimize a function of one distance metric subject to a budget constraint on the other distance metric. We present algorithms that provide solutions which are within a small constant factor of the objective value while violating the budget constraint by only a small constant factor.

Keywords: Location Theory, Approximation Algorithms, Computational Complexity, NP-hardness.

AMS(MOS) subject classification. 68R10, 68Q15, 68Q25.

[1] University of Würzburg, Am Hubland, 97074 Würzburg, Germany. Email: {krumke, noltemei}@informatik.uni-wuerzburg.de

[2] Los Alamos National Laboratory P.O. Box 1663, MS K990, Los Alamos NM 87545. Email: madhav@c3.lanl.gov. The work is supported by the Department of Energy under Contract W-7405-ENG-36.

[3] Part of the work was done when the author was at the University at Albany. Current Address: Mail-stop 47LA, Hewlett-Packard Company, 19447 Pruneridge Avenue, Cupertino, California 95014-9913. Email: rven@cup.hp.com.

[4] Department of Computer Science, University at Albany - SUNY, Albany, NY 12222. Email: {djr,ravi}@cs.albany.edu.

[5] Preliminary versions of parts of the paper appear in [RKM+93], [KN+95a] and [KN+95b].

1 Introduction and motivation

Several fundamental problems in location theory [HM79, MF90] involve finding a placement of facilities obeying certain "covering" constraints. Generally, the goal of such a location problem is to find a placement of minimum cost that satisfies all the specified constraints. In general, finding a placement of sufficient generality minimizing a cost measure is often NP-hard [GJ79]. Here, we consider several problems dealing with the placement of a specified number of facilities on the nodes of a given network so as to minimize some function of the distances between the facilities. Because of the nature of the placement desired, we refer to these problems as *compact location problems*. For example, consider the following processor allocation problem which arises in the context of multiprocessor systems. We are given a computational task consisting of a number of communicating subtasks. At a given time, some of the processors may already be allocated and the remaining processors are available. The problem is to select a subset of processors from the currently available set of processors, one processor per subtask, such that the cost of communication among the processors executing the subtasks is minimized. In this application, the processors must be allocated quickly, and this may conflict with the goal of minimum communication cost among the selected processors.

Such location problems are commonly modeled as problems on undirected graphs. The nodes of the graph represent the available sites. A cost is associated with each pair of sites, and it is specified as the weight of the edge joining the corresponding pair of nodes. Depending on the problem, this cost represents one of several parameters such as the cost of transporting components between the pair of sites, the cost of setting up a communication link between the pair of sites, the time required to communicate between the pair of sites, etc. In some problems, there is also a weight associated with each node. This node weight may reflect the cost of setting up a facility at the corresponding site.

Under this graph theoretic setting, a placement is a subset of nodes of a given cardinality. The cost of a placement is a problem-specific function of the weights of the nodes and edges in the subgraph induced by the placement. Examples of such cost functions are the sum of the weights of all the edges in the placement (which may reflect the total cost of setting up communication links between each pair of chosen sites), the maximum weight of an edge in the placement (i.e., the *bottleneck cost*, which may reflect the maximum time needed to communicate between any pair of chosen sites), etc. The goal of a compact location problem is to find a placement of minimum cost.

In practice, it is often the case that a minimum cost placement must be chosen subject to budget constraints on other cost measures. We also consider such constrained problems where the goal is to find a placement that minimizes one cost measure subject to a budget constraint

on another cost measure. Since these problems involve two cost functions, we refer to them as *bicriteria compact location problems*. As an example, consider once again the scenario presented above in the context of high performance computing. If the processors need to communicate with each other frequently to exchange data, then these data communication delays increase the time needed to complete a task. Further, the computation may require the processors to be synchronized often, thus adding to the time needed to complete the task. Therefore, it is desirable to select a subset of processors so that the total communication cost among the processors is minimized and the maximum delay due to synchronization during the computation does not exceed a given bound.

Such compact location problems also arise in a number of other applications such as allocation of manufacturing sites for the components of a system so as to minimize the cost of transporting components, distributing the activities of a project among geographically dispersed offices so as to minimize the transportation or communication cost among the offices, statistical clustering, pattern recognition, load-balancing in distributed systems, etc. (see [An72, AMO93, HM79, MF90, KN$^+$95a] and the references cited therein).

In graph theoretic terms, (bicriteria) compact location problems can be formalized as follows. Suppose we are given *two* weight-functions c, d on the edges of the network. (For example, the first weight function c may represent the cost of constructing an edge, and the second weight function d may represent the actual transportation- or communication-cost over an edge once it has been constructed.) Given such a graph, a positive number B and a positive integer p, we define a general bicriteria compact location problem $(\mathcal{A}, \mathcal{B})$ by identifying two minimization objectives of interest from a set of possible objectives. The parameter B represents the budget on the second objective $\mathcal{B}$ and the goal is to find a placement of p facilities having the minimum possible value for the first objective $\mathcal{A}$ such that this solution obeys the budget constraint on the second objective. For example, consider the *Diameter-Constrained Minimum Diameter Problem* denoted by (Min-Dia, Dia): Given an undirected complete graph $G = (V, E)$ with two nonnegative integral edge weight functions c (modeling the building cost) and d (modeling the delay or the communication cost), an integer p denoting the number of facilities to be placed, and an integral bound B (on the delay), find a placement of p facilities with minimum diameter under the c-cost such that the diameter of the placement under the d-costs (the maximum delay between any pair of nodes) is at most B.

The remainder of the paper is organized as follows. Section 2 contains preliminaries and the formal definitions of the problems considered in this paper. In Section 3 we summarize the results obtained. In Section 4 we discuss the related research done in this area. Section 5 contains non-approximability results for graphs with arbitrary weights. In Section 6 we present basic approximation algorithms for the unicriterion compact location problems. Section 7

contains our approximation algorithms for the diameter constrained bicriteria problems. In Section 8 we outline the approximation algorithms for the sum constrained problems. Section 9 discusses some extensions of our approximation algorithms. Finally, Section 10 contains concluding remarks and directions for future research.

2 Preliminaries and problem definitions

We consider a complete undirected n-vertex graph $G = (V, E)$ with one or two distance functions specified on the edges. Given an integer p, a *placement* P is a subset of V with $|P| = p$. Let δ denote a distance function on the edges of G. We use $\mathcal{D}_\delta(P) := \max_{\substack{e=(v,w) \\ v,w \in P}} \delta(v, w)$ to denote the *diameter*, $\mathcal{S}_\delta(P) := \sum_{\substack{e=(v,w) \\ v,w \in P}} \delta(v, w)$ to denote the *sum of the distances* and $\mathcal{Q}_\delta(P) := \sum_{\substack{e=(v,w) \\ v,w \in P}} \delta^2(v, w)$ to denote the *sum of the squares of the distances* between the nodes in the placement P.

Note that the average length and the variance[6] [AI+91] of an edge in a placement P are equal to $\frac{2}{p(p-1)}\mathcal{S}_\delta(P)$ and $\frac{2}{p(p-1)}\mathcal{Q}_\delta(P)$ respectively. Since these differ from the total length and the sum of the squared distances only by the respective scaling factors, finding a placement of minimum average length or minimum variance is equivalent to finding a placement of minimum total length and minimum sum of the squared distances respectively. We use this fact throughout this paper.

As is standard in the literature, we say that a nonnegative edge-weight function δ satisfies the *triangle inequality*, if we have $\delta(v, w) \leq \delta(v, u) + \delta(u, w)$ for all $v, w, u \in V$. We now define the problems studied in this paper beginning with unicriterion problems.

Definition 2.1 *An instance of the* **minimum diameter placement problem**, (MIN-DIA)*, is given by a complete graph $G = (V, E)$, a nonnegative edge-weight function c and an integer $2 \leq p \leq |V|$. The problem is to find a placement P (i.e., a vertex subset of size p), that minimizes $\mathcal{D}_c(P)$.*

The **minimum average distance placement**, (MIN-SUM), and **minimum variance placement** problem, (MIN-VAR), are defined analogously. We now extend the above definition to bicriteria compact location problems.

Definition 2.2 *An instance of a* diameter constrained minimum diameter placement problem (MIN-DIA, DIA)*is given by a complete graph $G = (V, E)$,* two *nonnegative edge-weight functions c and d, an integer $2 \leq p \leq |V|$ and a positive number B. The goal is to find a placement P which minimizes $\mathcal{D}_c(P)$ subject to the constraint $\mathcal{D}_d(P) \leq B$.*

[6] In [AI+91], the scaling factor used to define variance is $\frac{1}{p}$. We use $\frac{2}{p(p-1)}$ for reasons of uniformity.

The analogous versions for the various combinations of objectives $\mathcal{D}_\delta$, $\mathcal{S}_\delta$ and $\mathcal{Q}_\delta$ are defined similarly.

Given a problem Π, we use TI-Π to denote the problem Π restricted to graphs in which *both* the edge weight functions satisfy the triangle inequality. We will see that the triangle inequality plays an important role in determining the approximability of the compact location problems defined above.

One of the goals of our work is to find good approximation algorithms for several compact location problems introduced here. By an approximation algorithm (heuristic) we mean a *polynomial time* algorithm which produces feasible, but not necessarily optimal, solutions. A *relative* approximation algorithm guarantees a solution which is within a *multiplicative* constant K of the optimal value for every instance of the problem. The multiplicative constant K is referred to as the *performance guarantee* provided by the algorithm. This paper is concerned with the study of relative approximation algorithms for the placement problems defined above.

As shown in Section 5, unless $\mathsf{P} = \mathsf{NP}$, for several bicriteria problems considered here, it is not possible to strictly satisfy the constraint on the d-distances. This motivates the definition of a slightly relaxed version of the performance of an approximation algorithm. Formally, let Π be a bicriteria compact location problem. An (α, β)-*approximation algorithm* for Π (or an algorithm with a performance of (α, β)) is a polynomial-time algorithm, which for any instance of Π does one of the following:

(a) It produces a solution within α times the optimal value with respect to the first distance function c, violating the constraint with respect to the second distance function d by a factor of at most β.

(b) It returns the information that no feasible placement exists at all.

Notice that if there is no feasible placement but there is a placement violating the constraint by a factor of at most β, an (α, β)-approximation algorithm has the choice of performing either action (a) or (b).

We close this section with some basic definitions and an important observation. The *set of neighbors* of a vertex v in G, denoted by $N(v, G)$, is defined by $N(v, G) := \{w : (v, w) \in E\}$. The *degree* $\deg(v, G)$ of v in G is the number of vertices in $N(v, G)$. For a subset $V' \subseteq V$ of nodes, we denote by $G[V']$ the subgraph of G induced by V'. Given a graph $G = (V, E)$, the graph $G^2 = (V, E^2)$ is defined by $(u, v) \in E^2$ if and only if there is a path in G between u and v consisting of at most two edges. Following [HS86], the *bottleneck graph* $\text{bottleneck}(G, \delta, M)$ of $G = (V, E)$ with respect to δ and a bound M is defined by

$$\text{bottleneck}(G, \delta, M) := (V, E'), \text{ where } E' := \{e \in E : \delta(e) \leq M\}.$$

Observation 2.3 *Let H be a subgraph of the complete graph $G = (V, E)$ with edge-weights $c(e)$ $(e \in E)$ satisfying the triangle inequality. Then the weight of the heaviest edge in H^2 is at most twice the weight of the heaviest edge in H.* ■

3 Summary of results

Here, we study the complexity and approximability of a number of unicriterion and bicriteria compact location problems. One contribution of this paper is a general framework that leads to efficient approximation algorithms with provable performance guarantees for several compact location problems.

In Section 5, we show that the compact location problems studied in this paper are NP-hard. We establish hardness results for the unicriterion compact location problems (see Definition 2.1) which also extend, with appropriate modifications, to bicriteria versions. Next, we prove that, in general, obtaining placements that are near-optimal is NP-hard. For the bicriteria versions these negative results continue to hold, even when we allow the budget constraint on the second cost function to be violated by a constant factor and one of the cost functions satisfies the triangle inequality.

Given these hardness results, we focus our attention on graphs in which *both* the weight functions satisfy the triangle inequality. In the past, a substantial amount of work has been carried out in investigating the approximability of problems arising in network design and location theory when edge weights satisfy the triangle inequality (for example see [RSL77, HS86, RRT94]). We refer the reader to the paper by Bern and Eppstein [BE95] for a comprehensive survey of other geometric location problems. We obtain the following approximability results.

1. We provide an efficient generic method for approximating the unicriterion compact location problems. The procedure runs $O(n^2)$ in time. For the (Min-Dia) problem, the algorithm provides a performance guarantee of 2. We also observe that no polynomial time heuristic for the (Min-Dia) problem can provide a better performance guarantee unless P = NP. For the (Min-Sum) and (Min-Var) problems, our heuristics provide performance guarantees of $2 - 2/p$ and $4 - 6/p$ respectively.

2. We provide a polynomial time approximation algorithm for TI-(Min-Dia, Dia) with performance guarantee $(2, 2)$. Furthermore, we show that unless P $\neq$ NP, no polynomial time approximation algorithm can provide a performance guarantee of $(2 - \varepsilon, 2)$ or $(2, 2 - \varepsilon)$, for any $\varepsilon > 0$.

3. We give a polynomial time approximation algorithm for TI-(Min-Sum, Dia) with

performance guarantee $(2 - \frac{2}{p}, 2)$. We also show that, unless P $\neq$ NP, no polynomial time approximation algorithm can provide a performance guarantee of $(\alpha, 2 - \varepsilon)$, for any $\alpha \geq 1, \varepsilon > 0$.

4. We provide a polynomial time approximation algorithm for TI-(MIN-DIA, SUM) with performance guarantee $(2, 2 - \frac{2}{p})$. Furthermore, we show that unless P $\neq$ NP, no polynomial time approximation algorithm can provide a performance guarantee of $(2 - \varepsilon, \alpha)$, for any $\alpha \geq 1, \varepsilon > 0$.

5. For all $\gamma > 0$, we give a polynomial time approximation algorithm for TI-(MIN-SUM, SUM) with performance guarantee $((1+\gamma)(2-\frac{2}{p}), (1+\frac{1}{\gamma})(2-\frac{2}{p}))$. We also show that this algorithm can be implemented to run in time $O(n^2 \log n)$ using an elegant technique of Megiddo [Meg83].

All the above approximation results can be extended to the case when there weights on vertices. They can also be extended to the case where we are given a subset $D \subseteq V$ that must be included in the solution. We discuss these extensions in Section 9.

Our results are based on two basic techniques. The first is a combination of two ideas, namely the *power of graphs* approach of Hochbaum and Shmoys [HS86] and the local search approach for approximating single criteria compact location problems introduced in [KN+95a]. The second is an application of a *parametric search technique* similar to the one discussed in [MR+95] for network design problems.

4 Related work

In contrast to the above NP-hardness results which hold for general distance matrices, geometric versions of (MIN-DIA) and (MIN-VAR) were shown to be solvable in polynomial time in [AI+91]. In the geometric versions of these problems, the nodes are points in space and the distance between a pair of nodes is their Euclidean distance. For points in the plane, [AI+91] contains an $O(p^{2.5} n \log p + n \log n)$ algorithm for the (MIN-DIA) problem and an $O(p^2 n \log n)$ algorithm for the (MIN-VAR) problem, and it is observed that these algorithms extend to higher dimensions. These algorithms are based on the construction of p^{th} order Voronoi diagrams [Lee82, PS85].

Problems involving the placement of p facilities so as to minimize suitable cost measures have been studied extensively in the literature (see [BE95, AI+91, DL+93, KP93, EE94] and the references cited therein). These problems can roughly be divided into two main categories. The first category of problems involves selecting a set of p facilities so as to minimize (or maximize) the cost (distance) from the unselected sites to the selected sites. Problems that can be cast in this framework include the p-center problem [HS86, DF85],

the p-cluster problem [HS86, FG88, Go85] and the p-median problem [LV92, MF90]. The second category consists of problems where the goal is to select p facilities so as to optimize a certain cost measure defined on the set of selected facilities. Problems that can be cast in this framework include the p-dispersion problem [RRT94, Ta91, EN89], and the k-minimum spanning tree problem [RR$^+$94, GH94, AA$^+$94, BCV95, ZL93].

In contrast, not much work has been done in finding optimal location of facilities when there is more than one objective. A notable work in this direction is by Bar-Ilan, Kortsarz and Peleg [BKP93] who considered the problem of assigning network centers, with a bound imposed on the number of nodes that any center can service. We refer the reader to [MR$^+$95] for a survey of the work done in the area of algorithms for bicriteria network design and location theory problems.

5 Hardness results

In this section, we prove our non-approximability results. The hardness results are first established for the unicriterion versions of compact location problems. Then we show how these hardness results imply the hardness of bicriteria versions of compact location problems. Several of our hardness results use recent hardness results concerning approximability of the MAX-CLIQUE problem. We recall the definition and the related non-approximability result for the sake of completeness.

Definition 5.1 *An instance of the* CLIQUE *problem consists of an undirected graph* $G = (V, E)$ *and an integer* $J \leq |V|$. *The question is whether there exists a clique of size* J *in* G; *that is, whether there exist* J *vertices* $S = \{v_1, \ldots, v_J\}$ *such that* $\forall v_i, v_k \in S$ *we have* $(v_i, v_k) \in E$.

The optimization version of the problem, denoted by MAX-CLIQUE, *is to find a clique of maximum size in* G.

Recently, using a new characterization of NP in terms of probabilistically checkable proof systems, it has been shown that the MAX CLIQUE problem cannot be efficiently approximated. We state the most recent result obtained by Bellare and Sudan [BS94].

Theorem 5.2 *Unless* P = NP, *for all* $\varepsilon > 0$ *the problem* MAX-CLIQUE *does not have a polynomial time approximation algorithm with performance guarantee* $|V|^{1/6-\varepsilon}$. ■

5.1 Unicriterion Problems

Proposition 5.3 *The problems* (Min-Dia), (Min-Sum) *and* (Min-Var) *are* NP*-hard, even when the distances satisfy the triangle inequality.*

Proof: We will sketch the NP-hardness proofs for the (MIN-DIA) and (MIN-SUM) problems. The proof for (MIN-VAR) is similar.

We use a reduction from CLIQUE. Given an instance of the CLIQUE problem given by a graph $G = (V, E)$ and an integer J , construct an instance of (MIN-DIA) consisting of the complete graph on the node set V. For nodes x and y in V, let $c(x, y) = 1$ if $\{x, y\} \in E$ and let $c(x, y) = 2$ otherwise. Clearly, the distances satisfy the triangle inequality. It is straightforward to verify that G has a clique of size J if and only if we can place J facilities such that the diameter of the placement is equal to 1.

The same construction as before works for (MIN-SUM). If G has a clique of size J, then the nodes which form this clique constitute an optimal solution for the (MIN-SUM) instance of objective function value $J(J-1)/2$. If G does not have a clique of size J, then any placement of J nodes has cost at least $J(J-1)/2 + 1$. ■

The proof of the last proposition also shows that the existence of a polynomial time algorithm for TI-(MIN-DIA) with a performance of $2-\varepsilon$ for some $\varepsilon > 0$ implies that P = NP. If the distances are not required to satisfy the triangle inequality, we can replace the distance 2 in the above construction by any polynomial time computable function f. This shows that an approximation algorithm with a performance guarantee of $f(|V|)$ for any of the problems (MIN-DIA), (MIN-SUM) and (MIN-VAR) can be used to decide CLIQUE. We thus obtain the following results.

Proposition 5.4 *If the distances are not required to satisfy the triangle inequality, then unless* P = NP*, for any polynomial time computable function f there is no polynomial time approximation algorithm for any of the problems* (MIN-DIA)*,* (MIN-SUM)*,* (MIN-VAR) *with a performance of $f(|V|)$.*

For any $\varepsilon > 0$, TI-(MIN-DIA) *is* NP*-hard to approximate within a factor of $2 - \varepsilon$.* ■

The NP-hardness of the unicriterion compact location problems naturally implies the hardness of the bicriteria versions. Moreover, one can establish even stronger hardness results, as shown below.

Lemma 5.5 *Unless* P = NP*, for any $\varepsilon, \varepsilon' > 0$, there can be no polynomial time algorithm* A *which given an arbitrary instance of TI-*(MIN-DIA, DIA)*,*

- *either returns a subset $S \subseteq V$ of at least $\frac{2p}{|V|^{1/6-\varepsilon'}}$ nodes satisfying $\mathcal{D}_d(S) \leq (2-\varepsilon)B$,*
- *or provides the information that a placement of p nodes having d-diameter of at most B does not exist.*

Proof: Let I' be an arbitrary instance of MAX-CLIQUE, given by a graph $G' = (V', E')$. Without loss of generality we can assume that $E' \neq \emptyset$. We will give a many-one Turing reduction to the problem TI-(MIN-DIA, DIA).

For each $2 \leq k \leq |V|$, we construct an instance $I^{(k)}$ of TI-(MIN-DIA, DIA) as follows: Let $G^{(k)} = (V', E)$ $(E = \{(u, v) : u, v \in V, u \neq v\})$ and define $c^{(k)}, d^{(k)} : E \to \mathbb{N}$ by $c^{(k)}(e) := 1$ for all $e \in E$ and

$$d^{(k)}(e) := \begin{cases} 1 & \text{if } e \in E' \\ 2 & \text{otherwise.} \end{cases}$$

It is trivial to see that both weight functions obey the triangle inequality. We let $B^{(k)} := 1$ and $p^{(k)} := k$. Notice that the size of an instance $I^{(k)}$ is still polynomial in the size of I', and that we have constructed only polynomially many (namely, $O(|V|)$) instances. Now consider an instance $I^{(k)}$. Note that any placement P of $p^{(k)} = k$ nodes that has diameter $\mathcal{D}_{d^{(k)}}(P) \leq (2 - \varepsilon)B = (2 - \varepsilon)$ must have diameter 1 and, thus must form a clique in the original graph G'.

Assume that the original graph G' has a clique C of size $p^{(k)} = k$. Then this clique will satisfy $\mathcal{D}_{c^{(k)}}(C) = \mathcal{D}_{d^{(k)}}(C) = 1 = B^{(k)}$. By our assumption, the algorithm A must return a set S of at least $\frac{2p}{|V|^{1/6-\varepsilon'}}$ nodes with d-diameter at most $(2 - \varepsilon)B = (2 - \varepsilon) < 2$. Thus, as noted above, the algorithm must find a placement of diameter 1, and this set will form a clique in the original graph G'.

If there is no clique of size $p^{(k)} = k$ in G', any placement P of $p^{(k)} = k$ nodes in G' must include at least one edge e of length $d^{(k)}(e) = 2 > (2 - \varepsilon)$. Now, according to our assumptions about A, the algorithm has the choice of either returning a set of size at least $\frac{2p}{|V|^{1/6-\varepsilon'}}$ that will form a clique in the original graph or providing the information that there is no placement P of diameter at most $B = 1$.

Thus, the output of the algorithm A can be used to either obtain the information that G' does not contain a clique of size $p^{(k)} = k$ or that G' does have a clique of size at least $\frac{2p}{|V|^{1/6-\varepsilon'}}$.

Now, we run A for all the instances $I^{(k)}$ $(2 \leq k \leq |V|)$. Since the size of each instance $I^{(k)}$ is polynomial in the size of I' and we only have $O(|V|)$ instances, this will result in an overall polynomial time algorithm, according to our assumptions about A. Let $m := \max\{k : \text{A returns a set } S \text{ of diameter } 1\}$. Then, by our observations from above, we can conclude that G' has a clique of size at least $\frac{2m}{|V|^{1/6-\varepsilon'}}$ and that there is no clique of size $m + 1$ in G'. Hence, we can approximate the maximum clique number of G' by a factor of at most $\frac{m+1}{2m} \cdot |V|^{1/6-\varepsilon'} \leq |V|^{1/6-\varepsilon'}$. By the results in [BS94] (Theorem 5.2), this would imply that P = NP. ■

Again, replacing the factor 2 by a suitable polynomial time computable function f (e.g. $f = \Theta(2^{\text{poly}(|V|)})$, which given an input length of $\Omega(|V|)$ is polynomial time computable), it can be seen that, if the triangle inequality is not required to hold, there can be no polynomial time approximation with performance ratio $f(|V|)$ for either the optimal function value or the constraint (modulo $\mathsf{P} = \mathsf{NP}$). Thus, we obtain:

Theorem 5.6 *Unless* $\mathsf{P} = \mathsf{NP}$, *for all* $\varepsilon > 0, \varepsilon' > 0$ *the problem TI-*(MIN-DIA, DIA) *does not have a polynomial time approximation algorithm that finds a placement of at least* $2p/|V|^{1/6-\varepsilon'}$ *facilities and which has a performance guarantee of* $(\alpha, 2-\varepsilon)$ *or* $(2-\varepsilon, \beta)$.

Unless $\mathsf{P} = \mathsf{NP}$, *for any polynomial time computable functions* f, g, *if the c-costs and d-costs do not satisfy triangle inequality, then the problem* (MIN-DIA, DIA) *does not have a polynomial time* $(f(|V|), g(|V|))$*-approximation algorithm.* ■

The results of Lemma 5.5 also give a slightly stronger hardness result than the one stated in the previous theorem. Namely, we can conclude that, for any fixed $\alpha, \beta \geq 1$, unless *both* the c-costs and the d-costs satisfy the triangle inequality, we cannot obtain a polynomial time (α, β)-approximation algorithm for the problem (MIN-DIA, DIA). By essentially the same arguments, we obtain the following results.

Theorem 5.7 *Unless* $\mathsf{P} = \mathsf{NP}$, *for any fixed* $\alpha, \beta \geq 1$, *if the c-costs or the d-costs do not satisfy triangle inequality, the problems* (MIN-SUM, DIA), (MIN-DIA, SUM), *and* (MIN-SUM, SUM) *do not have polynomial time* (α, β)*-approximation algorithms.*

Unless $\mathsf{P} = \mathsf{NP}$, *for any fixed* $\alpha \geq 1$ *and* $\varepsilon > 0$, *the problem TI-*(MIN-SUM, DIA) *does not have a polynomial time* $(\alpha, 2-\varepsilon)$*-approximation algorithm.*

Unless $\mathsf{P} = \mathsf{NP}$, *for all fixed* $\varepsilon > 0$ *and* $\beta \geq 1$, *for the problem TI-*(MIN-DIA, SUM) *does not have a polynomial time* $(2-\varepsilon, \beta)$*-approximation algorithm.* ■

6 Unicriterion compact location problems

In this section, we present our approximation algorithms for the unicriterion versions of compact location problems. We begin (Section 6.1) by discussing our approximation algorithm for the minimum variance problem and then discuss (Section 6.2) similar heuristics for the minimum sum and minimum diameter problems. We also present (Section 6.3) a comparison of solutions under the three objectives, to demonstrate the orthogonal nature of the objectives.

procedure Gen-Alg$(G = (V, E), \mathcal{M}_c)$

Comment: $G = (V, E)$ denotes the graph with edge and/or node weights. $\mathcal{M}_c$ is a function that returns the cost of a given placement.

1. **if** $|V| < p$ **then return** "certificate of failure"
2. **else**
   begin
   (a) Solution $\leftarrow \emptyset$ and Value $\leftarrow \infty$.
   (b) **for** each node $v \in V$ **do**
   begin
   i. Find $N(v) = \{v_1, \ldots, v_{p-1}\} \subseteq V - \{v\}$ such that $N(v)$ contains $p-1$ nodes in $V - \{v\}$ closest to v.
   ii. $P(v) = N(v) \cup \{v\}$.
   iii. $t \leftarrow \mathcal{M}_c(P(v))$.
   iv. **if** $t <$ Value **then** Solution $\leftarrow P(v)$ and Value $\leftarrow t$.
   end
   end
3. **output** Solution and Value.

Figure 1: Generic Algorithm

6.1 Approximating minimum variance

Recall that the goal of (Min-Var) problem is to find a placement P for which $\mathcal{Q}_c(P)$, the sum of the squares of the distances between the nodes in the placement, is minimized.

Our heuristic for the (Min-Var) problem, denoted by Heur-(Min-Var), consists merely of a call to the generic procedure of Figure 1 with G as the graph corresponding to the problem instance and $\mathcal{M}_c := \hat{\mathcal{Q}}_c$, where for a placement $P(v) = \{v, v_1, \ldots, v_{p-1}\}$ consisting of a node v and its $p-1$ nearest neighbors $N(v) = \{v_1, \ldots, v_{p-1}\}$ with respect to c, the function $\hat{\mathcal{Q}}_c$ is defined by

$$\hat{\mathcal{Q}}_c(P(v)) := \hat{\mathcal{Q}}_c(\{v\} \cup N(v)) := \sum_{i=1}^{p-1} [c(v, v_i)]^2 . \tag{1}$$

Although the generic algorithm in Figure 1 is simple, it is quite powerful. We now establish the performance of the algorithm. This proof requires an extended form of the triangle inequality given by the following lemma.

Lemma 6.1 *If the distances in a network satisfy the triangle inequality, then for all nodes* x, y, z,

$$[c(x, z)]^2 \leq 2 \left([c(x, y)]^2 + [c(y, z)]^2 \right) .$$

Proof: By the triangle inequality, we have $c(x,z) \leq c(x,y) + c(y,z)$. By squaring both sides we obtain

$$[c(x,z)]^2 \leq [c(x,y)]^2 + [c(y,z)]^2 + 2c(x,y)c(y,z). \tag{2}$$

Moreover, we obtain from

$$0 \leq (c(x,y) - c(y,z))^2 = [c(x,y)]^2 + [c(y,z)]^2 - 2c(x,y)c(y,z) \tag{3}$$

that

$$2c(x,y)c(y,z) \leq [c(x,y)]^2 + [c(y,z)]^2 . \tag{4}$$

Thus by (2) and (4)

$$[c(x,z)]^2 \leq [c(x,y)]^2 + [c(y,z)]^2 + 2c(x,y)c(y,z) \leq 2\left([c(x,y)]^2 + [c(y,z)]^2\right) \tag{5}$$

which proves the lemma. ■

Lemma 6.2 *Let $v \in V$ be an arbitrary node and let $N(v) = \{w_1, \ldots, w_{p-1}\}$ be the set of nearest neighbors of v in $V - \{v\}$ with respect to c. Define*

$$Q_v := \hat{\mathcal{Q}}_c(\{v\} \cup N(v)) = \sum_{w \in N(v)} [c(v,w)]^2 . \tag{6}$$

Then $\mathcal{Q}_c(\{v\} \cup N(v)) \leq (2p-3)Q_v$.

Proof:

$$\begin{aligned}
\mathcal{Q}_c(\{v\} \cup N(v)) &= \sum_{w \in N(v) \cup \{v\}} [c(v,w)]^2 + \sum_{\substack{u,w \in N(v) \cup \{v\} \\ u \neq w}} [c(u,w)]^2 \\
&\leq Q_v + \sum_{\substack{u,w \in N(v) \cup \{v\} \\ u \neq w}} 2\left([c(u,v)]^2 + [c(v,w)]^2\right) \quad \text{(by Lemma 6.1)} \\
&= Q_v + \sum_{w \in N(v) \cup \{v\}} \sum_{u \in N(v) \cup \{v,w\}} 2\,[c(v,w)]^2 \\
&= Q_v + 2(p-2) \sum_{w \in N(v) \cup \{v\}} [c(v,w)]^2 \\
&= Q_v + 2(p-2)Q_v \\
&= (2p-3)Q_v
\end{aligned}$$

■

Lemma 6.3 *Let I be an instance of* (MIN-VAR). *Let $P^* \subseteq V$ be an optimal placement and let $P \subseteq V$ be the placement produced by* Heur-(MIN-VAR) *respectively for the instance I. Then $\mathcal{Q}_c(P)/\mathcal{Q}_c(P^*) \leq 4 - 6/p$.*

Proof: For each node $w \in P^*$, let

$$R_w = \sum_{v \in P^*} [c(v,w)]^2 . \tag{7}$$

We have

$$\mathcal{Q}_c(P^*) = \frac{1}{2} \sum_{w \in P^*} R_w. \tag{8}$$

Choose $v \in P^*$ such that

$$R_v = \min_{w \in P^*} R_w. \tag{9}$$

Then

$$\mathcal{Q}_c(P^*) = \frac{1}{2} \sum_{w \in P^*} R_w \geq \frac{1}{2} \sum_{w \in P^*} R_v = \frac{p}{2} R_v. \tag{10}$$

This yields

$$R_v \leq \frac{2}{p} \mathcal{Q}(P^*). \tag{11}$$

Consider the iteration of the for loop (Step 2(b) of Figure 1), where the node v chosen above is considered by the heuristic. Let $N(v) = \{w_1, \ldots, w_{p-1}\}$ be the set of nearest neighbors of v in $V - \{v\}$ chosen in that iteration. By definition of R_v and Q_v we have

$$Q_v = \sum_{w \in N(v)} [c(v,w)]^2 \leq \sum_{w \in P^*} [c(v,w)]^2 = R_v, \tag{12}$$

because $N(v)$ is chosen as the set of $p-1$ nearest neighbors of v. Then

$$\begin{aligned}
(2p-3)Q_v &\leq (2p-3)R_v \quad \text{(by Equation (12))} && (13)\\
&\leq 2\left[\frac{2p-3}{p}\right] \mathcal{Q}_c(P^*) \quad \text{(by Equation (11))} && (14)\\
&= \left[4 - \frac{6}{p}\right] \mathcal{Q}_c(P^*). && (15)
\end{aligned}$$

By construction, our algorithm will choose a node $\tilde{v}$ with minimum value $Q_{\tilde{v}} = \hat{\mathcal{Q}}_c(\{\tilde{v}\} \cup N(\tilde{v}))$. Thus $Q_{\tilde{v}} \leq Q_v$ and we get

$$\begin{aligned}
\mathcal{Q}(N(\tilde{v}) \cup \{\tilde{v}\}) &\leq (2p-3)Q_{\tilde{v}} \quad \text{(by Lemma 6.2)}\\
&\leq (2p-3)Q_v\\
&\leq \left[4 - \frac{6}{p}\right] \mathcal{Q}_c(P^*) \quad \text{(by Equation (15))}.
\end{aligned}$$

This completes the proof. ■

Running Time

We now address the running time of our algorithm. Let $n := |V|$ be the number of nodes in the graph G. Then the number m of edges is $m = \frac{n(n-1)}{2} = \Theta(n^2)$. The main effort of the algorithm is in the loop in Step 2(b). For each node v we must determine the $(p-1)$ nearest neighbors with respect to the c distance function. This can be done in $O(n+p)$ time as follows. We first use a linear time selection algorithm (see e.g. [CLR90]) to find the node w with the $(p-1)^{\text{st}}$ smallest distance from v. Then, by performing one linear pass over the $O(n)$ neighbors of v and comparing their distance to $c(v,w)$, we can then extract the $p-1$ nearest neighbors of v in $O(n+p)$ time.

Then, the algorithm computes the sum $\hat{\mathcal{Q}}_c(P(v)) = \sum_{i=1}^{p-1} [c(v,v_i)]^2$ for each of the n placements $P(v) = \{v, v_1, \ldots, v_{p-1}\}$. Evaluating $\hat{\mathcal{Q}}_c$ for one placement needs $O(p)$ time. Choosing the best placement $P(v)$ (with respect to $\hat{\mathcal{Q}}_c$) can then be accomplished in $O(n)$ time. This results in an overall time of $O(n^2 + np) = O(n^2)$.

Theorem 6.4 *The generic algorithm, called with $\mathcal{M}_c := \hat{\mathcal{Q}}_c$ is an approximation algorithm for* (MIN-VAR) *with a performance guarantee of $4 - 6/p$ and a running time of $O(n^2)$.* ■

6.2 Approximating TI-(MIN-SUM) and TI-(MIN-DIA)

We are going to establish two lemmas which enable us to use measures that can be computed faster than $\mathcal{S}_c$ and $\mathcal{D}_c$.

Lemma 6.5 *Let $v \in V$ be an arbitrary node and let $N(v) = \{w_1, \ldots, w_{p-1}\}$ be the set of nearest neighbors of v in $V - \{v\}$ with respect to c. Define*

$$S_v := \hat{\mathcal{S}}_c(\{v\} \cup N(v)) := \sum_{w \in N(v)} c(v,w). \tag{16}$$

and

$$D_v := \hat{\mathcal{D}}_c(\{v\} \cup N(v)) := \max_{w \in N(v)} c(v,w). \tag{17}$$

Then $\mathcal{S}_c(\{v\} \cup N(v)) \leq (p-1)S_v$ and $\mathcal{D}_c(\{v\} \cup N(v)) \leq 2D_v$.

Proof: We first prove the bound with respect to $\mathcal{S}_c$. Let $w \in N(v)$ be arbitrary. Then

$$\begin{aligned} \sum_{u \in N(v) \cup \{v\} \setminus \{w\}} c(w,u) &= c(w,v) + \sum_{u \in N(v) \setminus \{w\}} c(w,u) \\ &\leq c(w,v) + \sum_{u \in N(v) \setminus \{w\}} (c(w,v) + c(v,u)) \end{aligned}$$

$$
\begin{aligned}
&= (p-1)c(w,v) + \sum_{u \in N(v)\setminus\{w\}} c(v,u) \\
&= (p-2)c(v,w) + \sum_{u \in N(v)} c(v,u) \\
&= (p-2)c(v,w) + S_v. \qquad (18)
\end{aligned}
$$

Now using (18) we obtain

$$
\begin{aligned}
\mathcal{S}_c(P(v)) &= \mathcal{S}_c(N(v) \cup \{v\}) \qquad (19) \\
&= \frac{1}{2}\left(\sum_{u \in N(v)} c(v,u) + \sum_{w \in N(v)} \sum_{u \in N(v)\cup\{v\}\setminus\{w\}} c(w,u) \right) \\
&= \frac{1}{2} S_v + \frac{1}{2} \sum_{w \in N(v)} \sum_{u \in N(v)\cup\{v\}\setminus\{w\}} c(w,u) \\
&\overset{(18)}{\leq} \frac{1}{2} S_v + \frac{1}{2} \sum_{w \in N(v)} ((p-2)c(v,w) + S_v) \\
&= \frac{1}{2} S_v + \frac{p-2}{2} S_v + \frac{p-1}{2} S_v \\
&= (p-1) S_v.
\end{aligned}
$$

This proves the first part of the lemma. For the second part, observe that $P(v)$ will form a clique in the square of the bottleneck graph bottleneck(G, c, D_v). The claim now follows immediately from Observation 2.3. ■

Using the above lemma and following a proof outline similar to those of Lemma 6.5 and Lemma 6.3, we obtain:

Theorem 6.6 *Denote by* Heur-(MIN-SUM) *and* Heur-(MIN-DIA) *the algorithms resulting by setting* $\mathcal{M}_c := \hat{\mathcal{S}}_c$ *and* $\mathcal{M}_c := \hat{\mathcal{D}}_c$ *respectively in the generic procedure shown in Figure 1.*

Then Heur-(MIN-SUM) *as applied to TI-*(MIN-SUM) *has a performance of* $2 - 2/p$. Heur-(MIN-DIA) *is an approximation algorithm for TI-*(MIN-DIA)*for with a performance of* 2. *Both algorithms have running time* $O(n^2)$. ■

We will prove more general versions of Theorem 6.6 in Sections 7 and 8.

6.3 Comparisons of solutions under the three objectives

We presented heuristics for the unicriterion compact location problems, which produce near-optimal solutions under the three objectives, namely minimum diameter, minimum average

distance and minimum variance. This section is devoted to a comparison of the three objectives. We present problem instances to show that a solution which is optimal with respect to one objective can be arbitrarily poor with respect to another objective, even when the triangle inequality holds. These examples demonstrate that the objectives are in some sense orthogonal.

Our examples use suitable combinations of the following sets of nodes with distance functions. (The "combination" operation will be specified after the definitions of these sets.) The distance functions are symmetric and it is easy to verify that they satisfy the triangle inequality. In the following specifications, ϵ is an arbitrarily small positive quantity.

1. Let $X = \{x_1, x_2, \ldots, x_p\}$ be a set of nodes with $c(x_i, x_j) = 1$, $1 \leq i < j \leq p-1$.
2. Let $Y = \{y_0, y_1, y_2, \ldots, y_{p-1}\}$ with $c(y_0, y_i) = 1 + \epsilon$ for $1 \leq i \leq (p-1)$ and $c(y_i, y_j) = \epsilon$ for $1 \leq i < j \leq (p-1)$.
3. Let $Z = \{z_0, z_1, z_2, \ldots, z_{p-1}\}$ with $c(z_0, z_i) = (p-1)/2$ for $1 \leq i \leq (p-1)$ and $c(z_i, z_j) = \epsilon$ for $1 \leq i < j \leq (p-1)$.
4. Let $R = \{r_0, r_1, \ldots, r_{p-1}\}$ with $c(r_0, r_i) = \sqrt{p/2 - 1}$ for $1 \leq i \leq (p-1)$ and $c(r_i, r_j) = \epsilon$ for $1 \leq i < j \leq (p-1)$.
5. Let $W = \{w_0, w_1, w_2, \ldots, w_{p-1}\}$ with $c(w_0, w_i) = p/2 - 1$ for $1 \leq i \leq (p-1)$ and $c(w_i, w_j) = \epsilon$ for $1 \leq i < j \leq (p-1)$.
6. Let $T = \{t_0, t_1, t_2, \ldots, t_{p-1}\}$ with $c(t_0, t_i) = \sqrt{p/2 + 1}$ for $1 \leq i \leq (p-1)$ and $c(t_i, t_j) = \epsilon$ for $1 \leq i < j \leq (p-1)$.

For each set α defined above, Table 1 gives the values of $\mathcal{D}_c(\alpha)$, $\mathcal{S}_c(\alpha)$ and $\mathcal{Q}_c(\alpha)$. We will refer to this table several times in the following discussion.

For disjoint sets of nodes α and β with distance functions c_α and c_β respectively, define $\mathsf{COMBINE}(\alpha, \beta)$ as the graph with the node set $\alpha \cup \beta$ and the distance function $c_{\alpha\cup\beta}$ given by

$$\begin{array}{rcll} c_{\alpha\cup\beta}(x,y) & = & c_\alpha(x,y) & \text{if } x, y \in \alpha \\ & = & c_\beta(x,y) & \text{if } x, y \in \beta \\ & = & M & \text{if } x \in \alpha \text{ and } y \in \beta, \end{array}$$

where M is a quantity which is much larger than any of the distances in α and β. It is easy to verify that if c_α and c_β satisfy the triangle inequality, then so does $c_{\alpha\cup\beta}$.

We use the sets and the $\mathsf{COMBINE}$ operation defined above to create instances that bring out the orthogonality of the measures. The reader should bear in mind that in each case, the number of facilities to be placed is p and that ϵ can be made arbitrarily small.

Set α	$\mathcal{D}_c(\alpha)$	$\mathcal{S}_c(\alpha)$	$\mathcal{Q}_c(\alpha)$
X	1	$p(p-1)/2$	$p(p-1)/2$
Y	$1+\epsilon$	$(p-1)((1+\epsilon)+(p-2)\epsilon/2)$	$(p-1)((1+\epsilon)^2+(p-2)\epsilon^2/2)$
Z	$(p-1)/2$	$(p-1)((p-1)/2+(p-2)\epsilon/2)$	$(p-1)((p-1)^2/4+(p-2)\epsilon^2/2)$
R	$\sqrt{p/2-1}$	$(p-1)(\sqrt{p/2-1}+(p-2)\epsilon/2)$	$(p-1)((p/2-1)+(p-2)\epsilon^2/2)$
W	$p/2-1$	$(p-1)((p/2-1)+(p-2)\epsilon/2)$	$(p-1)((p/2-1)^2+(p-2)\epsilon^2/2)$
T	$\sqrt{p/2+1}$	$(p-1)(\sqrt{p/2+1}+(p-2)\epsilon/2)$	$(p-1)((p/2+1)+(p-2)\epsilon^2/2)$

Table 1: Comparison of the Three Objectives

1. Diameter and Sum of the distances:
   Consider the graph $\mathsf{COMBINE}(X, Y)$. For this graph, the set X is the optimal placement with respect to diameter while the set Y is the optimal placement with respect to sum of the distances. From Table 1, it is easy to see that $\mathcal{S}_c(X)$ is arbitrarily worse than $\mathcal{S}_c(Y)$. Thus, for this graph, the set which is optimal with respect to diameter is arbitrarily poor with respect to sum of the distances.

2. Diameter and Sum of the squares of the distances:
   Here also the graph $\mathsf{COMBINE}(X, Y)$ suffices because X is optimal with respect to diameter and Y is optimal with respect to sum of the squares of the distances. From Table 1, it is seen that $\mathcal{Q}_c(X)$ is arbitrarily worse than $\mathcal{Q}_c(Y)$.

3. Sum of the distances and Diameter:
   For this case, we consider the graph $\mathsf{COMBINE}(X, Z)$. The set Z minimizes the sum of the distances while X minimizes the diameter. Observe from Table 1 that $\mathcal{D}_c(Z)$ is arbitrarily worse than $\mathcal{D}_c(X)$.

4. Sum of the squares of the distances and Diameter:
   Consider the graph $\mathsf{COMBINE}(X, R)$. The set R minimizes the sum of the squares of the distances while X minimizes the diameter. Observe from Table 1 that $\mathcal{D}_c(R)$ is arbitrarily worse than $\mathcal{D}_c(X)$.

5. Sum of the distances and Sum of the squares of the distances:
   Consider the graph $\mathsf{COMBINE}(X, W)$. The set W minimizes the sum of the distances

while X minimizes the sum of the squares of the distances. Observe from Table 1 that $\mathcal{Q}_c(W)$ is arbitrarily worse than $\mathcal{Q}_c(X)$.

6. Sum of the squares of the distances and Sum of the distances:
   Consider the graph $\mathsf{COMBINE}(X, T)$. The set X minimizes the sum of the squares of the distances while T minimizes the sum of the distances. Observe from Table 1 that $\mathcal{S}_c(X)$ is arbitrarily worse than $\mathcal{S}_c(T)$.

7 Heuristics for diameter constrained problems

In this section, we discuss our approximation algorithms for bicriteria compact location problems when the budget constraint is on the diameter. We begin with a brief discussion of the basic technique (Section 7.1). This is followed by our approximation algorithms for TI-(Min-Sum, Dia) (Section 7.2) and TI-(Min-Dia, Dia) (Section 7.3).

7.1 Basic technique

We discuss the basic technique used in approximating the diameter constrained compact location problems. Hochbaum and Shmoys [HS86] introduced a general approach for approximating a number of bottleneck problems when the costs satisfy triangle inequality. We combine this approach with the local search heuristic for compact location problem developed in [KN+95a] to obtain our approximation algorithms for diameter constrained compact location problems.

To illustrate our ideas, consider the problem TI-(Min-Sum, Dia). Let B be the bound on the diameter of the placement with respect to the d-cost. It is clear that in the subgraph induced by an optimal placement no edge has d-cost more than B. Thus we can prune all the edges in the graph with d-costs more than B. Call the resulting graph G'. Since finding an optimal placement that minimizes the the sum of distances with respect to c-cost is "hard" even in the modified graph, we resort to finding a near optimal placement. We find a near optimal placement in G' with respect to the c-costs. This placement will turn into a clique in the square graph $(G')^2$. Because the edge weights satisfy triangle inequality, it follows that the placement does not violate the constraint by a factor of more than 2.

7.2 Approximating TI-(Min-Sum, Dia)

We present an approximation algorithm for the problem TI-(Min-Sum, Dia) that provides a performance guarantee of $(2 - 2/p, 2)$. The algorithm is shown in Figure 2, where the cost measure $\mathcal{M}_c$ corresponds to $\hat{\mathcal{S}}_c$, defined in (16).

Procedure HEUR-FOR-DIA-CONSTRAINT

1 $G' := \text{bottleneck}(G, d, B)$
2 $V_{cand} := \{v \in G' : \deg(v, G') \geq p - 1\}$
3 **if** $V_{cand} = \emptyset$ **then return** "certificate of failure"
4 Let $best := +\infty$
5 Let $P_{best} := \emptyset$
6 **for** each $v \in V_{cand}$ **do**
7 Let $N(v, G') = \{w_1, \ldots, w_r\}$ with $c(v, w_1) \leq \cdots \leq c(v, w_r)$
8 Let $P(v) := \{v, w_1, \ldots, w_{p-1}\}$
9 **if** $\mathcal{M}_c(P(v)) < best$ **then** $P_{best} := P(v)$
10 **output** P_{best}

Figure 2: Details of the heuristic TI-(Min-Sum, Dia).

Performance guarantee

We will show that a placement returned by the algorithm will be almost feasible in the sense that it violates the diameter constraint by a factor of at most 2.

Lemma 7.1 *Any placement considered by the algorithm in Step 8 has d-diameter at most* $2B$.

Proof: Observe that any placement $P(v)$ which is considered by the algorithm consists of a vertex v and $p - 1$ vertices $w_1, \ldots, w_{p-1}$ that are adjacent to v in the bottleneck graph $G' := \text{bottleneck}(G, d, B)$, which does not contain edges of d-weight greater than B. The placement will form a clique a clique in $(G')^2$. By Observation 2.3 all the edges between the vertices in this clique have d-weight at most $2B$. In other words, the d-diameter of $P(v)$ is at most $2B$. ■

As an immediate consequence of Lemma 7.1 we obtain the following corollary.

Corollary 7.2 *If the algorithm returns a placement P (i.e., the algorithm does not report that that no feasible solution exists), then the d-diameter of P is at most* $2B$. ■

We are now ready to establish the performance of the algorithm in Figure 2.

Lemma 7.3 *Algorithm* HEUR-FOR-DIA-CONSTRAINT *called with $\mathcal{M}_c := \hat{\mathcal{S}}_c$ has a performance of* $(2 - 2/p, 2)$.

Proof: By Corollary 7.2 we know that any solution output by the algorithm will violate the constraint on the d-diameter by a factor of at most 2. Assume that the algorithm informs

about the infeasibility of an instance in Step 3. Then, indeed, no feasible solution to the instance can exist, since any feasible solution with d-diameter at most B will form a clique in the bottleneck graph G'. In particular there will be at least p nodes of degree p or greater in G' and V_{cand} can not be empty.

It remains to show that for any instance of TI-(MIN-SUM, DIA) with a nonempty set of feasible solutions, the algorithm will find a good placement with respect to the objective $\mathcal{S}_c$.

Consider an optimal solution P^* such that $\mathcal{D}_d(P^*) \leq B$ and let $OPT := \mathcal{S}_c(P^*)$ be the optimal objective function value. By definition, this placement forms a clique of size p in $G' := \text{bottleneck}(G, d, B)$. Hence, for any node $v \in P^*$ we have $|N(v, G')| \geq p$ and $P^* \subseteq V_{cand}$.

Define for each node $v \in P^*$:

$$R_v := \sum_{\substack{w \in P^* \\ w \neq v}} c(v, w).$$

We have $\mathcal{S}_c(P^*) = \frac{1}{2} \sum_{v \in P^*} R_v$. Now let $v \in P^*$ be so that R_v is a minimum among all nodes in P^*. Then clearly

$$OPT = \mathcal{S}_c(P^*) \geq \frac{p}{2} R_v. \tag{20}$$

As mentioned earlier, $v \in V_{cand}$. Consider the step of the algorithm in which it examines v. Let $N(v) := P(v) \setminus \{v\}$ denote the set of $p-1$ nearest neighbors of v in G' with respect to c. Then we have

$$S_v := \sum_{\substack{w \in N(v) \\ w \neq v}} c(v, w) \leq R_v \leq \frac{2}{p} OPT. \tag{21}$$

by definition of $N(v)$ as the set of nearest neighbors. By Lemma 6.5 we have

$$\begin{aligned} \mathcal{S}_c(P(v)) &\leq (p-1) S_v \\ &\overset{(21)}{\leq} (2 - 2/p) OPT. \end{aligned}$$

As the algorithm chooses the placement P_{best} with the least value of $\hat{\mathcal{S}}_c$, the claimed performance guarantee follows. ■

Running Time

Let $n := |V|$ be the number of nodes in the graph G. Then the number m of edges is $m = \frac{n(n-1)}{2} = \Theta(n^2)$. Thus, the bottleneck graph G' in Step 1 of the algorithm can be computed in time $O(m)$ by simply inspecting the weight $d(e)$ for each edge $e \in E$. The set of candidate nodes V_{cand} can then be determined in $O(n)$ time. Now, using the same analysis as in Section 6 for the generic algorithm, it now follows that the total time complexity of

Algorithm HEUR-FOR-DIA-CONSTRAINT is $O(n^2 + np) = O(n^2)$. We summarize our results in the following theorem.

Theorem 7.4 *Algorithm* HEUR-FOR-DIA-CONSTRAINT *called with* $\mathcal{M}_c := \hat{\mathcal{S}}_c$ *is an approximation algorithm for TI-*(MIN-SUM, DIA) *with a performance of* $(2 - 2/p, 2)$ *and a running time of* $O(n^2)$. ■

Lower bound example

We provide an example of an instance I of TI-(MIN-SUM, DIA), where the $(2 - 2/p, 2)$-performance of our heuristic is asymptotically tight. In this example, the node set V consists of $p+1$ subsets $B_0, \ldots, B_p$. The set $B_0 = \{v_1, \ldots, v_p\}$ forms an optimal placement of p nodes and the distances between the nodes in B_0 are $c(v_i, v_j) = 1$ and $d(v_i, v_j) = 1$. All other sets B_i $(i > 1)$ consist of $p - 1$ nodes and we have that $c(u, w) = 2 - 2\varepsilon$ and $d(u, w) = 2$ $(u, w \in B_i)$.

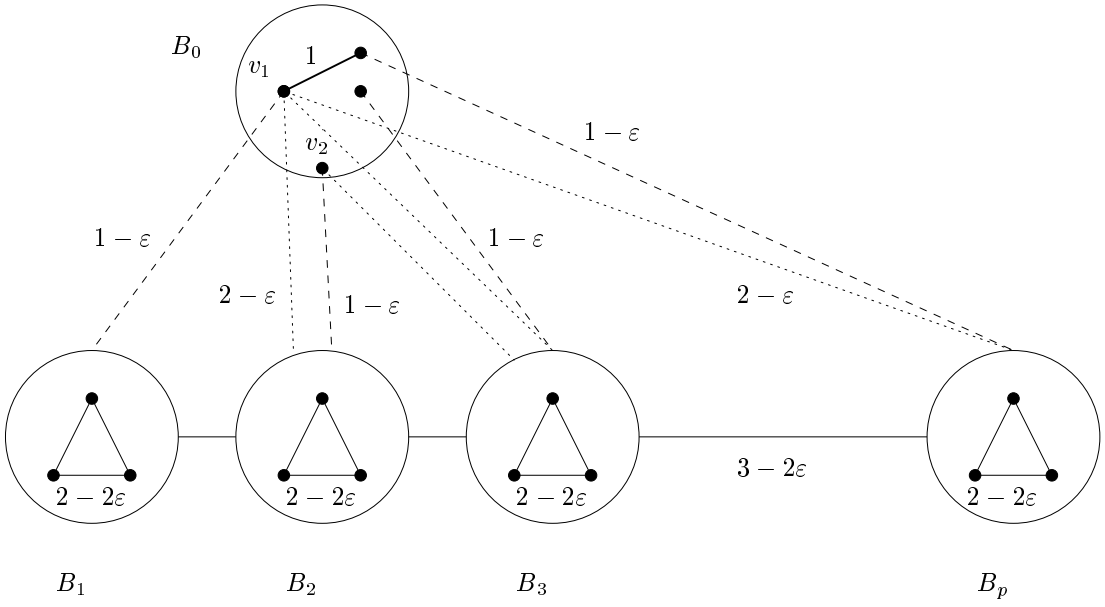

Figure 3: Lower Bound Example for the Heuristic for TI-(MIN-SUM, DIA)

Let v_i be any node in the optimum placement B_0. we set $c(v_i, w) = 1 - \varepsilon$ and $d(v_i, w) = 1$ for all $w \in B_i$, while we let $c(v_i, w') = 2 - \varepsilon$ and $d(v_i, w) = 2$ for all $w' \in B_j, j \neq i$.

Finally, for each edge (u, v) such that u and v belong to distinct sets B_i, we set the $c(u, v) = 3 - 2\varepsilon$ and $d(u, v) = 2$. It is easily verified that the distances c and d obey the triangle inequality. The instance I just described is illustrated in Figure 3.

For the bound $B := 1$, the set B_0 forms an optimum placement with objective function value $\mathcal{S}_c(B_0) = \frac{p(p-1)}{2}$ and diameter $\mathcal{D}_d(B_0) = 1$. In the first step, the algorithm calculates the bottleneck graph $G' = \text{bottleneck}(G, d, 1)$. This graph now only contains the edges between the nodes in B_0 and the edges of the form (v_i, w) with $w \in B_i$. It is now easy to see that $V_{cand} = B_0$. For each node $v_i \in B_0$ the set of $p-1$ nearest neighbors consists of the set B_i and $\mathcal{D}_d(\{v_i\} \cup B_i) = 2 = 2B$. Moreover $\mathcal{S}_c(\{v_i\} \cup B_i) = (p-1)(1-\varepsilon) + \frac{(p-1)(p-2)}{2}(2-2\varepsilon)$ and $\lim_{\varepsilon \to 0} \frac{\mathcal{S}_c(\{v_i\} \cup B_i)}{\mathcal{S}_c(B_0)} = 2 - 2/p$.

7.3 Approximating TI-(Min-Dia, Dia)

Using the results from Section 6 in conjunction with the results in [MR+95] we can devise an approximation algorithm with a performance guarantee $(4, 4)$ for TI-(Min-Dia, Dia). Here, we present an improved heuristic for this problem. This heuristic provides a performance guarantee of $(2, 2)$. In view of Theorem 5.6, this is the best approximation we can expect to obtain in polynomial time.

The heuristic for approximating TI-(Min-Dia, Dia) is the same as HEUR-FOR-DIA-CONSTRAINT given in Figure 2, except that the measure $\mathcal{M}_c := \hat{\mathcal{D}}_c$, where $\hat{\mathcal{D}}_c$ is defined in (17).

Performance guarantee and running time

Theorem 7.5 *Algorithm* HEUR-FOR-DIA-CONSTRAINT *called with* $\mathcal{M}_c := \hat{\mathcal{D}}_c$ *(where* $\hat{\mathcal{D}}_c$ *is defined in (17)) is an approximation algorithm for TI-*(Min-Sum, Dia) *with a performance of* $(2, 2)$. ■

The proof of Theorem 7.5 is deferred until Section 9, where we will state and prove an extension of this result (Theorem 9.3). It should be noted that the same running time analysis as in the case of TI-(Min-Sum, Dia) shows that the algorithm runs in time $O(n^2)$.

Lower bound example

The lower bound example in Figure 3 can be used to show that the approximation ratio indicated in Theorem 7.5 is tight for the heuristic. Again, the optimum placement consists of the nodes in B_0 which has both c- and d-cost equal to 1. The placement returned by the heuristic is one of the sets $\{v_i\} \cup B_i$, which has c-cost equal to $2 - 2\varepsilon$ and d-cost equal to 2.

8 Heuristics for sum constrained problems

In this section, we study approximation algorithms for bicriteria compact location problems where the objective is to minimize either the diameter $\mathcal{D}_c$ or the sum of the distances $\mathcal{S}_c$ subject to constraints of sum type.

8.1 Approximating TI-(Min-Dia, Sum)

An (α, β)-approximation algorithm for TI-(Min-Dia, Sum) can be constructed with the help of a (β, α)-approximation algorithm A for TI-(Min-Dia, Dia) in the following way. Given an instance of TI-(Min-Dia, Sum) with the bound B on the sum of the d-distances we do the following: We find the minimum $M \in \{\ c(e) : e \in E\ \}$ such that A as applied to the instance I' of TI-(Min-Sum, Dia) with the bound B' on the c-diameter set to M finds a set of nodes where the total sum of the distances between the nodes is at most βB.

After sorting the set $\{\ c(e) : e \in E\ \}$ in $O(m \log m)$ time, where $m := |E| = \frac{n(n-1)}{2}$, we can use binary search and obtain an approximation algorithm with a performance of (β, α) for TI-(Min-Dia, Sum). This algorithm issues $O(\log m) = O(\log n)$ calls to A.

Thus, taking into account our results from Section 7.2 we obtain an approximation algorithm for TI-(Min-Sum, Sum) with the properties stated in the following theorem.

Theorem 8.1 *There is an approximation algorithm for TI-*(Min-Dia, Sum) *with performance of* $(2, 2 - 2/p)$ *and running time* $O(n^2 \log n)$. ■

Lower bound example

As a lower bound example take again the instance depicted in Figure 3 but interchange the c- and d-costs and set the bound B to be $\frac{p(p-1)}{2}$. It is easy to see that by choosing $\varepsilon > 0$ small enough, the performance for this instance is arbitrarily close to $(2, 2 - 2/p)$.

8.2 Approximating TI-(Min-Sum, Sum)

We proceed to present a heuristic for TI-(Min-Sum, Sum). The basic idea behind the approximation algorithm is to use a *parametric search* technique to reduce the problem to that of solving the minimum sum problem for a modified weight function. Then, by appropriate scaling and rounding techniques, this solution can be transformed back into a near optimal solution for the original bicriteria problem.

The presentation of our approximation algorithm is organized as follows. We first present our heuristic and show that it provides a performance guarantee of $((2 - 2/p)(1 + 1/\gamma), (2 - 2/p)(1 + \gamma))$ for any fixed $\gamma > 0$. We then show how to improve the running time of the heuristic using an elegant technique due to Megiddo [Meg83].

Procedure HEUR-FOR-(MIN-SUM, SUM)

1 Use a binary search to find the smallest integer

$$M \in \mathcal{I} := \left[0, p^2 \max\{\ c(e) : e \in E\ \}\right]$$

such that Sum-Test(T)=Yes.

2 **if** the binary search terminates with the information that there is no such integer **then output** "There is no feasible solution" and **stop**

3 Let P be placement generated by Sum-Test(T)

4 **if** $\mathcal{S}_d(P) > (2-2/p)B$ **then output** "There is no feasible solution" **else output** P

Figure 4: Main procedure for TI-(MIN-SUM, SUM).

Procedure Sum-Test(M)

1 Let $\mu := \frac{M}{B}$.

2 **for** each pair (v, w) of nodes define the distance function $h(v, w)$ by $h(v, w) := c(v, w) + \mu d(v, w)$.

3 Compute a $(2-2/p)$-approximation for the (MIN-SUM) instance given by the graph G the number p and distances $h(e)$, $e \in E$

4 Let P_M be a set of p nodes with $\mathcal{S}_h(P_M) \leq (2-2/p) \cdot \min\limits_{\substack{P \subseteq V \\ |P|=p}} \mathcal{S}_h(P)$.

5 **if** $\mathcal{S}_h(P_M) \leq (2-2/p)(1+\gamma)M$ **then output** "Yes" **else output** "No".

Figure 5: Test procedure used for TI-(MIN-SUM, SUM).

8.2.1 A flexible but slow heuristic

The main procedure shown in Figure 4 uses the test procedure from Figure 5. Step 3 of the main procedure, that is, computing a $(2-2/p)$-approximation for the constructed instance of (MIN-SUM) (i.e. minimizing $\mathcal{S}_h$), can be done by using the unicriterion algorithm from Section 6 for (MIN-SUM) (or the algorithm from Section 7.2, where we set $c := h$, $d := 1$ and $B := 1$). We also note that γ is a fixed quantity that specifies the accuracy requirement.

For a value of M let OPT_{h_M} denote the sum of the distances of an optimal placement of p nodes with respect to the distance function $h_M(v, w) := c(v, w) + \frac{M}{B} d(v, w) = c(v, w) + \mu d(v, w)$; that is

$$OPT_{h_M} = \min_{\substack{P \subseteq V \\ |P|=p}} \mathcal{S}_{h_M}(P).$$

Then we have the following lemma:

Lemma 8.2 *The function $F(M) = \frac{OPT_{h_M}}{M}$ is monotonically nonincreasing on for $M > 0$.*

Proof: Let M_1 and M_2 be given numbers with $M_1 < M_2$. Let P_1 and P_2 denote optimal placements of p nodes under h_M when $M = M_1$ and $M = M_2$ respectively. For $i \in \{1, 2\}$,

let C_i and D_i denote the costs of placement P_i under c and d respectively. Thus, we have that $F(M_i) = \frac{C_i}{M_i} + \frac{D_i}{B}$ for $i \in \{1, 2\}$.

Consider the cost under h of the placement P_1 when $M = M_2$. By the definition of C_1 and D_1, it follows that the cost of P_1 is $C_1 + \frac{D_1 \cdot M_2}{B}$. Thus the value of $F(M_2)$ is at most this cost divided by M_2 which is $\frac{C_1}{M_2} + \frac{D_1}{B}$. This in turn is less than $\frac{C_1}{M_1} + \frac{D_1}{B}$, since $M_1 < M_2$. But $\frac{C_1}{M_1} + \frac{D_1}{B}$ is exactly $F(M_1)$, and hence $F(M_1) \geq F(M_2)$. ■

The next corollary proves that the binary search in Algorithm HEUR-FOR-(MIN-SUM, SUM) can work correctly. Before we state and prove the corollary, observe that Step 4 of the algorithm ensures that, if the algorithm outputs a placement, this placement will violate the constraint on the sum of the d-distances by a factor of at most $2 - 2/p$. Thus, in the sequel we can restrict ourselves to instances with nonempty set of feasible solutions. Given such an instance, let $OPT = \mathcal{S}_c(P^*)$ denote the function value of an optimal placement P^* of p nodes. To simplify the analysis, we assume that OPT/γ is an integer. This can be enforced by first scaling the cost function c so that all values are integers and then scaling again by γ.

Corollary 8.3 *The test procedure* Sum-Test *returns "*Yes*" for all $M > OPT/\gamma$. Thus, the binary search in Algorithm* HEUR-FOR-(MIN-SUM, SUM) *works correctly and either finds a value $M' \leq OPT/\gamma$ or provides the information that* Sum-Test *returns "*No*" for all values of M.*

Proof: We first show that the procedure will return "Yes" if called with $M^* = OPT/\gamma$. Notice that M^* is an integer by our assumption. We estimate the sum of the h_{M^*}-distances between the nodes in the optimal placement P^*. This sum is then $OPT + \frac{M^*}{B}B = OPT + M^* = (1+\gamma)M^*$. Thus it follows that $OPT_{h_{M^*}} \leq (1+\gamma)B^*$ and the $(2-2/p)$-approximation P_T that is computed in Step 3 will satisfy $\mathcal{S}_h(P_{M^*}) \leq (2-2/p)OPT_{h_{M^*}} \leq (2-2/p)(1+\gamma)M^*$.

Thus, we observe that the procedure will return "Yes". Moreover, since $OPT_{h_{M^*}} \leq (1+\gamma)B^*$, it follows that $F(M^*) \leq (1+\gamma)$, where F is the function defined in Lemma 8.2. By the results of this lemma we then have $F(M) \leq (1+\gamma)$ for all $M \geq M^*$.

This is equivalent to saying that for all $M \geq M^*$ the corresponding optimal placement P^*_M minimizing $\mathcal{S}_{h_M}$ satisfies $\mathcal{S}_{h_M}(P^*_M) \leq (1+\gamma)M$. Hence for all these values of M, the approximation P_M computed in Step 3 of the test procedure will satisfy $\mathcal{S}_{h_M}(P_M) \leq (2-2/p)(1+\gamma)M$. But this means that Algorithm 5 will return "Yes" for all $M \geq M^*$. ■

Now we are ready to complete the proof of the performance of our approximation algorithm.

Lemma 8.4 *For any fixed $\gamma > 0$ Algorithm* HEUR-FOR-(MIN-SUM, SUM)*, as applied to TI-*(MIN-SUM, SUM)*, has a performance of $((2-2/p)(1+1/\gamma), (2-2/p)(1+\gamma))$.*

Proof: By Corollary 8.3, the binary search in Algorithm 4 will successfully end with a value of M satisfying $M \leq OPT/\gamma$. Let P_M be the corresponding placement that is returned by Sum-Test. Then we have

$$\mathcal{S}_c(P_M) \leq \mathcal{S}_h(P_M) \leq (2 - \frac{2}{p})(OPT + \frac{M}{B}B) \leq (2 - \frac{2}{p})(1 + \frac{1}{\gamma}) \cdot OPT.$$

Moreover, we see that

$$\frac{M}{B}\mathcal{S}_d(P_M) \leq \mathcal{S}_h(P_M) \leq (2 - \frac{2}{p})(1 + \gamma)M.$$

Multiplying the last chain of inequalities by B/M yields

$$\mathcal{S}_d(P_M) \leq (2 - 2/p)(1 + \gamma)B.$$

This completes the proof. ■

In the above version of the heuristic for TI-(MIN-SUM, SUM) the the test procedure Sum-Test is called $O\left(\log\left(\frac{p^2 c_{max}}{\gamma}\right)\right)$ times during the binary search, where $c_{max} := \max\{\, c(e) : e \in E \,\}$. For the rest of this section, let $T_{test}(n) = O(n^2)$ the time required for a single call to Sum-Test. Then, the total time for the algorithm would be $O\left(\log \frac{p^2 c_{max}}{\gamma} \cdot T_{test}(n)\right)$. We will now show how to improve this running time.

8.2.2 Outline of a faster heuristic

If Sum-Test is called with some parameter K, it first computes the compound weights h_K. Then, in Step 3 it computes a $2 - 2/p$-approximation for the (unicriterion) TI-(MIN-SUM) instance with edge weights given by h_K. This is done with the help of our algorithm from Section 6. Recall that this algorithm generated n placements $P(v)$, one for each vertex and its nearest neighbors with respect to h_K. It then outputs the placement with the best objective function value $\hat{\mathcal{S}}_{h_K}(P(v))$.

As before, $K^* \in \mathcal{I} := [0, p^2 \max\{\, c(e) : e \in E \,\}]$ be the minimum value such that Sum-Test(K^*) ="Yes". Assume that we already know the ordering of the edges in the graph with respect to h_{K^*}. Then, for each vertex v we can find the $p-1$ nearest neighbors with respect to h_{K^*}. We do *not* need to know the weights; the ordering suffices. Thus, given the knowledge about the ordering with respect to h_{K^*}, we can find a set of n placements containing the placement output by our slow heuristic for TI-(MIN-SUM, SUM) above.

This is what our faster algorithm will do in the first phase. It will find the ordering with respect to h_{K^*} and narrow our search to n placements. This is done *without* actually knowing K^*. In the second phase, we will determine the placement among these n placements which our slow heuristic would output.

8.2.3 Finding an ordering with respect to h_{K^*}

Let $m = \frac{n(n-1)}{2}$ be the number of edges in the graph $G = (V, E)$. Basically we wish to sort the set $S := \{h_{K^*}(e_1), \ldots, h_{K^*}(e_m)\}$ where K^* is not known. However, for any K we can decide whether $K^* \leq K$ or $K^* > K$ by one call to our test procedure Sum-Test: If Sum-Test(K) ="Yes", then we know that $K^* \leq K$. Otherwise, we can conclude that $K^* > K$.

Recall that the h_K-weight of an edge e is given by $c(e) + M\frac{d(e)}{B}$. Thus, for each edge e, the compound weight $h_K(e)$ viewed as a function of K is a linear function. Given two edges e and e', their ordering with respect to the compound weights h_K changes at most once when K varies, namely at the point where the two linear functions intersect. Clearly, given two edges e and e' this value of K can be computed in constant time.

To simplify the presentation, we will first sketch the main idea before going into details. Imagine applying a (sequential) sorting algorithm to S. The sorting algorithm would start by comparing some values $h_{K^*}(e)$ and $h_{K^*}(e')$. Then, we could do the the following:

We compute intersection point of the two linear functions $h_K(e)$ and $h_K(e')$. If they do not intersect, the ordering of e and e' will not change if K varies. Hence, inspecting $h_K(e)$ and $h_K(e')$ for *any* value of K will tell us whether $h_{K^*}(e) \geq h_{K^*}(e')$ or vice versa.. Otherwise, let $K_{e,e'}$ be the value of K where $h_K(e)$ and $h_K(e')$ intersect. We call our procedure Sum-Test for $K = K_{e,e'}$ and find out whether whether $K^* \leq K_{e,e'}$ or $K^* > K_{e,e'}$. Since the ordering of the edges e and e' only changes at $K_{e,e'}$, we can then decide whether $h_{K^*}(e) \geq h_{K^*}(e')$ or vice versa. Thus, by $O(1)$ calls to Sum-Test, we can determine the result of a comparison.

Using the idea from above in conjunction with a standard sequential sorting algorithm (which makes $O(m \log m)$ comparisons), we could find the ordering of the edges at K^* by $O(m \log m)$ calls to Sum-Test. However, using Megiddo's technique [Meg83] we can speed up the algorithm substantially.

The idea is to use an adaptation of a sequentialized parallel sorting algorithm such as Cole's scheme [Col88]. Recall that a comparison essentially consists of a call to Sum-Test computation, so comparisons are expensive. Using the parallel sorting scheme, we basically accept a greater total number of comparisons, but we can use the parallelism to group the independent comparisons made in one stage of the parallel machine and then answer all of them together efficiently.

Cole's algorithm uses m processors to sort an array of m elements in parallel time $O(\log m)$. Recall that in our case $m = |E|$ is the number of edges in the graph $G = (V, E)$. The algorithm is simulated serially, employing one "processor" at a time, according to some fixed permutation, letting each perform one step in each cycle. When two values $h_{K^*}(e)$ and $h_{K^*}(e')$ have to be compared, we compute the intersection point of the two linear functions, where the ordering changes (but we do *not* know the result of the comparison at this point).

The crucial observation is that these critical values can be computed independently, meaning that each of the "processors" does not need any knowledge about the critical points computed by the other ones.

After the first of the $O(\log m)$ stages, we are given at most m critical values of K, say $K_1 \leq K_2 \leq \cdots \leq K_r$ with $r \leq m$. For convenience set $K_0 := -\infty$ and $K_{r+1} := +\infty$. Using binary search, we find an interval $[K_i, K_{i+1}]$, where K^* must be contained.

This is done in the following way: Start with $low := -\infty$ and $high := +\infty$. Then compute the median $M := K_{\lfloor (r+1)/2 \rfloor}$ of the K_j in $O(r)$ time. We then decide whether $K^* \leq M$ by one call to Sum-Test. If Sum-Test(K) ="Yes", then we know that $K^* \leq M$. Otherwise, $K^* > M$. In the first case, we set $high := M$ and remove all values K_j with $K_j > M$ from our set of critical values. Similarly, in the second case we set $low := M$ and remove the values smaller than the median M. Clearly, this can be done in $O(r)$ time. Since M was the median of the K_j the number of critical values decreases by a factor of one half.

Then, the total time effort $Time(r)$ for the binary search satisfies the recurrence:

$$Time(r) = Time(r/2) + T_{test}(n) + O(r),$$

where $T_{test}((n,p)$ is the time needed for one call to Sum-Test. The solution of the recurrence is $Time(r) = O(r + T_{test}(n) \log r)$. Since $r \in O(m)$, this shows that we obtain the interval $[K_i, K_{i+1}]$ containing K^* by $O(\log m)$ calls to Sum-Test plus an overhead of $O(m)$ elementary operations.

Notice that by construction the interval $[K_i, K_{i+1}]$ does not contain any critical points in the interior. If $K_i = K_{i+1}$, then we know that $K^* = K_i = K_{i+1}$. This way we have determined K^*. In this case we can compute the order of all edges with respect to h_{K^*} in $O(m \log m)$ time and stop the modified sorting algorithm.

Otherwise, the interior of $[K_i, K_{i+1}]$ is nonempty. By one call to Sum-Test we can find out whether $K^* \leq K_i$, which implies that $K^* = K_i$ since we know that $K^* \in [K_i, K_{i+1}]$. Again, the adopted sorting procedure can stop after having computed the ordering of all edges with respect to h_{K^*}.

The remaining case is that $K_i < K^* \leq K_{i+1}$. In this case it is easy to see that answering the comparisons from the first round by inspecting the weights h_τ, where $\tau \in (K_i, K_{i+1})$ is any interior point of the interval $[K_i, K_{i+1}]$, gives the same results as answering the comparisons with respect to the h_{K^*}-weights.

Thus, at the end of the the first round, our algorithm has either found K^* and thus the ordering of *all* edges in the graph with respect to their h_{K^*}-weights, or we can answer the comparisons from the first round using the ordering of the edges with respect to h_τ.

The above process is repeated $O(\log m)$ times, once for each parallel step of the parallel sorting machine. Since in each of the $O(\log m)$ rounds we answer all comparisons of the

parallel sorting scheme, upon termination we have found the ordering of the edges with respect to the h_{K^*}-weights.

The time needed for the first stage of the algorithm above can be estimated as follows: There are $O(\log m)$ cycles altogether. In each round we evaluate $O(m)$ intersection points. Also, we need $O(\log m)$ calls to Sum-Test plus the overhead of $O(m)$. Thus we have the following lemma:

Lemma 8.5 *The improved heuristic computes the ordering of the edges with respect to h_{K^*} in time $O(n^2 \log n)$.* ■

8.2.4 Finding the right placement

Let P_{slow} be the placement generated by our slow heuristic for TI-(Min-Sum, Sum). We have already argued that, given the ordering of the edges with respect to h_{K^*} we can find a set $\mathcal{P} = \{P(v_1), \ldots, P(v_n)\}$ of placements such that $P_{slow} \in \mathcal{P}$. By the construction of our slow algorithm, it follows that P_{slow} is a placement $P(v_j)$ in $\mathcal{P}$ with minimum $\hat{\mathcal{S}}_{h_{K^*}}(P(v_j))$.

Using the running time analysis from Section 7, we see that $\mathcal{P}$ can be determined in $O(n^2 + np \log n)$ time. We now show how to find P_{slow} in the set $\mathcal{P}$ efficiently.

For each placement $P(v_i) \in \mathcal{P}$ denote by $C_i := \hat{\mathcal{S}}_c(P(v_i))$ and $D_i := \hat{\mathcal{S}}_d(P(v_i))$ the simplified sum of the c-weights and d-weights respectively (see Equation (16)). Clearly, all the C_i and D_i can be found in an overall time of $O(np)$.

Observe that $\hat{\mathcal{S}}_{h_K}(P(v_i)) = C_i + K\frac{D_i}{B}$. Our task now becomes to find the placement having minimum value when $K = K^*$. Again, we can view $\hat{\mathcal{S}}_{h_K}(P(v_i))$ as a linear function of K. The ordering of two placement changes, when two of the functions intersect. Using the same technique as finding the ordering of the *edges* with respect to h_{K^*}, we can find the ordering of the *placements* in $\mathcal{P}$ with respect to $\hat{\mathcal{S}}_{h_{K^*}}$ in time $O(m \log m + T_{test}(n))$.

This enables us to find the same placement as the slow heuristic in an overall time of $O(m \log m + T_{test}(n) \log^2 m)$. Since, $T_{test}(n) = O(n^2)$ and $m = O(n^2)$, we obtain the following theorem.

Theorem 8.6 *For any fixed $\gamma > 0$, the improved heuristic for TI-*(Min-Sum, Sum) *has a performance of $((2 - 2/p)(1 + 1/\gamma), (2 - 2/p)(1 + \gamma))$ and a running time of $O(n^2 \log^2 n)$.*

9 Extensions

We now briefly discuss some extensions of our algorithms presented in the previous sections.

9.1 Extension to the node-weighted case

We show how to extend HEUR-FOR-DIA-CONSTRAINT to apply to the case when we additionally have a weight for each node in G, and the minimization objective is a function of both the edge weights and node weights. For the sake of brevity, we will illustrate our ideas for such an extension by considering one specific problem, namely TI-(MIN-SUM, DIA). The approximation algorithms for the other problems can be extended in a similar fashion.

The input consists again of a complete undirected graph $G = (V, E)$ with edge-weight functions c, d and weights $\omega(v)$, for each $v \in V$. The goal is to find a placement P, with $|P| = p$, such that the objective function

$$\mathcal{S}_c^{wt}(P) = \sum_{\substack{e=(v,w) \\ v,w \in P}} c(v,w) + \sum_{v \in P} \omega(v)$$

is minimized subject to the same constraint as for the (MIN-SUM, DIA) problem, namely $\mathcal{D}_d(P) \leq B$. We denote this extension of the (MIN-SUM, DIA) problem by (MIN-SUM, DIA)^{wt}.

To obtain an approximate solution for TI-(MIN-SUM, DIA)^{wt}, we transform a given instance I of TI-(MIN-SUM, DIA)^{wt} into an instance I' of TI-(MIN-SUM, DIA) as discussed in the proof of the following lemma.

Lemma 9.1 *Given any instance I of TI-(MIN-SUM, DIA)^{wt}, there are linear time transformations f and g such that both of the following conditions hold.*

1. *The transformation f constructs an instance I' of TI-(MIN-SUM, DIA) from I.*
2. *Given a placement P' of p nodes for I' with diameter D and objective function value C, g constructs a placement P of p nodes for I such that P has diameter D and objective function value C.*

Proof: Given the instance I, the transformation f constructs an instance I' of TI-(MIN-SUM, DIA) as follows. The nodes in $G'(V', E')$ are in one-to-one correspondence with the nodes of G. The d'-cost on each edge in G' is the same as the d-cost of the corresponding edge in G. The c'-cost for an edge (v', w') in G' is defined as follows:

$$c'(v', w') := c(v, w) + \frac{1}{2p}(\omega(v) + \omega(w)).$$

It is easy to check that the triangle inequality is satisfied for c'. Next consider a placement P' of p nodes in I'. The transformation g chooses as its placement P the p nodes in I

corresponding to the p nodes in I'. By a straightforward calculation, it can be seen that for any placement P of p nodes

$$\sum_{\substack{e'=(v',w') \\ v',w' \in P'}} c'(v',w') = \sum_{v \in P} \omega(v) + \sum_{\substack{e=(v,w) \\ v,w \in P}} c(v,w).$$

■

Combining Lemma 9.1 with Theorem 8.1, we obtain a $(2, 2-2/p)$-approximation for TI-(MIN-SUM, DIA)^{wt}.

9.2 Extension to distinguished nodes

We now turn to the extension of our techniques to another variant of the problems. In this variant (referred to as placement with *distinguished nodes*), we are given a subset $D \subseteq V$ that must be included in the solution. The objective function to be minimized becomes then $\mathcal{M}_c(P \cup D)$, where $\mathcal{M}_c \in \{\mathcal{D}_c, \mathcal{S}_c, \mathcal{Q}_c\}$. The problem is to find a set $P \subseteq V - D$ of p nodes minimizing $\mathcal{M}_c$ (subject to the constraint $\tilde{\mathcal{M}}_d(P \cup D) \leq B$, where again $\tilde{\mathcal{M}}_d \in \{\mathcal{D}_d, \mathcal{S}_d, \mathcal{Q}_d\}$).

As in the node weighted case, such extensions can be done for all the problems considered. For a problem Π, we use Π^D to denote the problem with distinguished nodes. Since the basic approach for solving each of the problems Π^D is similar, we illustrate our ideas by considering two problems, namely TI-(MIN-SUM, DIA)^D and TI-(MIN-DIA, DIA)^D. We have chosen these two problems since one of them involves two similar cost measures while the other involves two different cost measures.

Approximating TI-(MIN-SUM, DIA)^D

We show how to adopt HEUR-FOR-DIA-CONSTRAINT in Figure 2 to devise approximation algorithm for TI-(MIN-SUM, DIA)^D. The modified heuristic HEUR-DISTINGUISHED is given in Figure 6.

We now establish the performance guarantee of the modified heuristic. As the proof parallels the one for Theorem 8.1, we just sketch the main steps.

Theorem 9.2 *Algorithm* HEUR-DISTINGUISHED *called with* $\mathcal{M}_c = \mathcal{S}_c$ *is an approximation algorithm for TI-*(MIN-SUM, DIA)^D *with a performance of* $(2 - 2/(p + |D|), 2)$.

Proof: Consider an optimal solution P^* such that $\mathcal{D}_d(P^* \cup D) \leq B$. It can be seen that any node from $P^* \cup D$ is in V_{cand} and thus the heuristic will not output a certificate of failure. The fact that the diameter constraint $\mathcal{D}_d(P \cup D) \leq B$ is violated by a factor of at most 2 is again a consequence of the fact that any placement $P(v) \cup D$ considered by the heuristic forms a clique in $(G')^2$.

Procedure HEUR-DISTINGUISHED

$G' := \text{bottleneck}(G, d, B)$
$V_{cand} := \{v \in G' : v \notin D \text{ and } |N(v, G') \setminus D| \geq p-1\} \cup \{v \in D : |N(v, G') \setminus D| \geq p\}$
if $V_{cand} = \emptyset$ **then return** "certificate of failure"
Let $best := +\infty$
Let $P_{best} := \emptyset$
for each $v \in V_{cand}$ **do**
    Let $N_D(v, G') = \{w_1, \ldots, w_r\}$ with $c(v, w_1) \leq \cdots \leq c(v, w_r)$
    if $v \in D$ **then** let $P(v) := \{v, w_1, \ldots, w_p\}$ **else** let $P(v) := \{v, w_1, \ldots, w_{p-1}\}$
    if $\mathcal{M}_c(P(v) \cup D) < best$ **then** $P_{best} := P(v)$
output P_{best}

Figure 6: Modified Heuristic for TI-(MIN-SUM, DIA)^D.

Defining $\tilde{S}_v := \sum_{\substack{w \in P^* \cup D \\ w \neq v}} c(v, w)$ $(v \in P^* \cup D)$ and choosing $v \in P^* \cup D$ with minimum $\tilde{S}_v$, we have

$$\mathcal{S}_c(P^* \cup D) = \frac{1}{2} \sum_{v \in P^* \cup D} \tilde{S}_v \geq \frac{p + |D|}{2} \tilde{S}_v.$$

Consider the iteration of the main loop when the heuristic considers v. Let $N(v) := P(v) \setminus \{v\} = \{w_1, \ldots, w_k\}$ be the set of nearest neighbors of v in $V - D$, where $k = p$ if $v \in D$ and $k = p - 1$ if $v \in P^*$. Again, we have $S'_v := \sum_{w \in (P(v) \cup D) \setminus \{v\}} c(v, w) \leq \tilde{S}_v$, by the fact that $N(v)$ is the set of k nearest neighbors of v. Now,

$$\begin{aligned}
\mathcal{S}_c(P(v) \cup D) &= S'_v + \frac{1}{2} \sum_{u \in (P(v) \cup D) - \{v\}} \sum_{w \in (P(v) \cup D) - \{v,u\}} c(u, w) \\
&\leq S'_v + \frac{1}{2} \sum_{u \in (P(v) \cup D) - \{v\}} \sum_{w \in (P(v) \cup D) - \{v,u\}} (c(v, u) + c(v, w)) \\
&= S'_v + \frac{1}{2} \sum_{u \in (P(v) \cup D) - \{v\}} \left[\sum_{w \in (P(v) \cup D) - \{v,u\}} c(v, u) + \sum_{w \in (P(v) \cup D) - \{v,u\}} c(v, w) \right] \\
&= (p + |D| - 1) S'_v \\
&\leq (p + |D| - 1) \frac{2}{p + |D|} \cdot OPT(I) = \left(2 - \frac{2}{p + |D|}\right) \cdot OPT(I).
\end{aligned}$$

■

Approximating TI-(MIN-DIA, DIA)^D

As in the case of TI-(MIN-DIA, DIA), where we simply used the heuristic designed for TI-(MIN-SUM, DIA) with the measure $\mathcal{M}_c := \hat{\mathcal{D}}_c$, we can again use the heuristic HEUR-DISTINGUISHED for TI-(MIN-SUM, DIA)^D and merely set $\mathcal{M}_c := \hat{\mathcal{D}}_c^D$, where for a placement

$P(v)$ consisting of a node v and its r nearest neighbors $v_1, \ldots, v_r$, $(r \in \{p, p-1\})$

$$\hat{\mathcal{D}}_c^D(P(v)) := \max\{\ c(v,w) : w \in \{v_1, \ldots, v_r\} \cup D\ \} \tag{22}$$

We then obtain the following extension of Theorem 7.5.

Theorem 9.3 *Algorithm* HEUR-DISTINGUISHED*, called with* $\mathcal{M}_c := \hat{\mathcal{D}}_c^D$*, is an approximation algorithm for TI-*(MIN-DIA, DIA)^D *with a performance of* $(2,2)$.

Proof: Consider an optimal solution P^* such that $OPT := \mathcal{D}_d(P^* \cup D) \leq B$. It can be seen that $P^* \cup D \subseteq V_{cand}$, and thus V_{cand} is non-empty. Consequently, the heuristic will not output a "certificate of failure".

Any placement considered in the heuristic HEUR-DISTINGUISHED will turn into a clique in $(G')^2$, where $G' = \text{bottleneck}(G, d, B)$. Thus we conclude that any placement considered by the heuristic will violate the diameter constraint by a factor of at most 2.

Now consider an arbitrary node $v \in P^*$. Clearly $v \in V_{cand}$. Consider the step of the algorithm HEUR-DISTINGUISHED in which it considers v. It is easy to see that $\hat{\mathcal{D}}_c^D(\{v\} \cup \{w_1, \ldots, w_{p-1}\}) \leq OPT$. Thus, for the placement $P(\tilde{v})$ output by the algorithm $\hat{\mathcal{D}}_c^D(P(\tilde{v})) \leq OPT$. By the triangle inequality it follows that $\mathcal{D}_c(P(v) \cup D) \leq 2 \cdot \hat{\mathcal{D}}_c^D(P(\tilde{v})) \leq 2 \cdot OPT$. ■

The approximation algorithms for TI-(MIN-DIA, SUM) and TI-(MIN-SUM, SUM) can also be extended to handle the case of distinguished nodes. By using the ideas in this section and the techniques in Section 7, we obtain the following theorems.

Theorem 9.4 *There are polynomial time approximation algorithms for TI-*(MIN-DIA, SUM)^D *and TI-*(MIN-SUM, SUM)^D *with performances* $(2, 2-2/(p+|D|))$ *and* $((1+1/\gamma)(2-2/(p+|D|)), (1+\gamma)(2-2/p+|D|)))$*, respectively, where* $\gamma > 0$ *is any fixed constant.*

10 Concluding remarks

We introduced and studied the complexity and approximability of several natural bicriteria compact location problems. Our results demonstrate that when distance functions obey the triangle inequality, the problems are provably easier to approximate.

Tables 2 to 4 summarize our results. Table 2 shows the hardness results for the various unicriterion problems. Table 3 gives the corresponding approximation results. Table 4 shows our results for bicriteria compact location problems. The horizontal entries denote the objective function. For example the entry in row i, column j denotes the performance guarantee for the problem of minimizing objective j with a budget on objective i.

The results in this paper raise the following questions.

Problem	(MIN-DIA)	(MIN-SUM)	(MIN-VAR)
General	NP-hard (Prop. 5.3)	NP-hard (Prop. 5.3)	NP-hard (Prop. 5.3)
Triangle Inequality	NP-hard (Prop. 5.3)	NP-hard (Prop. 5.3)	NP-hard (Prop. 5.3)
1D Version	Efficiently solvable [AI+91]	Efficiently solvable	Efficiently solvable [AI+91]
2D Version	Efficiently solvable [AI+91]	Open	Efficiently solvable [AI+91]

Table 2: Complexity Results for Compact Location Problems

Note: The 1D version of (MIN-SUM) can be solved efficiently because every optimal solution consists of p contiguous points.

	(MIN-DIA)		(MIN-SUM)		(MIN-VAR)	
Problem	UB	LB	UB	LB	UB	LB
General	—	NGR (Prop. 5.4)	—	NGR (Prop. 5.4)	—	NGR (Prop. 5.4)
Triangle Inequality	2 (Thm. 6.6)	2 (Prop. 5.4)	2- $1/p$ (Thm. 6.4)	Open	4 - $6/p$ (Thm. 6.6)	Open

Table 3: Approximability Results for NP-hard Compact Location Problems

Notes: UB denotes the best known upper bound on the performance guarantee. LB denotes the lower bound on the performance guarantee; that is, the intrinsic limit assuming $\mathsf{P} \neq \mathsf{NP}$. NGR is an abbreviation for "No Guaranteed Ratio".

$\rightarrow$ Objective $\downarrow$ Budget	Diameter	Sum
Diameter	approximable within $(2, 2)$ not approximable within $(2-\varepsilon, 2)$ or $(2, 2-\varepsilon)$	approximable within $(2-\frac{2}{p}, 2)$ not approximable within $(\alpha, 2-\varepsilon)$
Sum	approximable within $(2, 2-\frac{2}{p})$ not approximable within $(2-\varepsilon, \alpha)$	approximable within $((1+\gamma)(2-\frac{2}{p}), (1+\frac{1}{\gamma})(2-\frac{2}{p}))$

Table 4: Performance guarantee results for bicriteria compact location in a complete graph with both the edge weights obeying the triangle inequality. $\gamma > 0$ is a fixed accuracy parameter. The non-approximability results stated assume that $\mathsf{P} \neq \mathsf{NP}$. As discussed in Section 9, these results can be extended to handle node weights and also the case of distinguished nodes.

1. Reference [RRT94] studies approximation algorithms for dispersion problems. It would be of interest to investigate approximation algorithms for bicriteria versions of practical dispersion problems. A slightly different bicriteria formulation involving node and edge weights for dispersion problems is considered in [RRT95].

2. In a companion paper [KN$^+$95c], we study bicriteria compact location problems when the underlying graph is a tree. It is of interest to consider other restricted graph classes which arise in the context of parallel computing and devise efficient algorithms for the bicriteria problems studied here. The n-dimensional hypercubes and n-dimensional meshes are two examples of natural graph classes for which bicriteria compact location problems could be investigated.

3. The compact location problem is an instance of a more general class of problems where the goal is to find an embedding of the problem topology (graph) onto a host computer topology so as to minimize the total communication cost. Placements produced using the algorithms presented in this paper may lead to heavy congestion in one or more of the links. Therefore, it is of interest to model and investigate compact location problems taking the congestion of links into account.

Acknowledgments: We thank Dr. Gyan Bhanot (Thinking Machines Corporation) for bringing to our attention the processor allocation problem mentioned in the introduction. We thank Professor R. Ravi (Carnegie-Mellon University) and Ravi Sundaram (Delta Trading, Inc.) for valuable discussions which lead to the research described in this paper. We also thank the referee for many helpful suggestions.

References

[AI+91] A. Aggarwal, H. Imai, N. Katoh, and S. Suri, Finding k points with Minimum Diameter and Related Problems. *J. Algorithms*, 12(1):38–56, March 1991.

[AMO93] R. K. Ahuja, T. L. Magnanti and J. B. Orlin, *Network Flows: Theory, Algorithms and Applications*, Prentice-Hall, Englewood Cliffs, NJ, 1993.

[An72] H. C. Andrews, *Introduction to Mathematical Techniques in Pattern Recognition*, Wiley-Interscience, New York, NY, 1972.

[AA+94] B. Awerbuch, Y. Azar, A. Blum, and S. Vempala, Improved approximation guarantees for minimum-weight k-trees and prize-collecting salesmen. *Proceedings of the 27th Annual ACM Symposium on the Theory of Computing* (STOC'95), pp. 277–376.

[BCV95] A. Blum, P. Chalasani and S. Vempala, A Constant-Factor Approximation for the k-MST Problem in the Plane. *Proceedings of the 27th Annual ACM Symposium on the Theory of Computing* (STOC'95), pp. 294–302.

[BKP93] J. Bar-Ilan, G. Kortsarz and D. Peleg, How to Allocate Network Centers. *J. Algorithms*, Vol. 15, No. 3, Nov. 1993, pp 385-415.

[BS94] M. Bellare and M. Sudan, Improved Non–Approximability Results. in *Proceedings of the 26th annual ACM Symposium on the Theory of Computing (STOC'94)*, May 1994, pp 184–193.

[BE95] M. Bern and D. Eppstein, Approximation Algorithms for Geometric Problems. Unpublished manuscript, January 1995.

[CLR90] T. H. Cormen, C. E. Leiserson, and R. L. Rivest, *Introduction to algorithms*, MIT Press, 1990.

[Col88] R. Cole, *Parallel merge sort*, SIAM Journal on Computing **17** (1988), no. 4, 770–785.

[DF85] M. E. Dyer and A. M. Frieze, A Simple Heuristic for the p-Center Problem. *Operations Research Letters*, 3(6):285–288, Feb. 1985.

[DL+93] A. Datta, H. P. Lenhof, C. Schwarz and M. Smid, Static and dynamic algorithms for k-point clustering problems. In *Algorithms and Data Structures, Third Workshop*, LNCS Volume 709, Springer Verlag, August, 1993, Pages 265–276.

[EE94] J. Erickson and D. Eppstein, Iterated nearest neighbors and finding minimal polytopes. *Discrete and Computational Geometry*, (11) 1994, pp. 321–350.

[EN89] E. Erkut and S. Neuman, Analytical Models for Locating Undesirable Facilities. *European J. Operations Research*, 40:275–291, 1989.

[FG88] T. Feder and D. Greene, Optimal Algorithms for Approximate Clustering. In *ACM Symposium on Theory of Computing (STOC'88)*, May 1988, pp 434–444.

[GH94] N. Garg and D. S. Hochbaum, An $O(\log k)$ Approximation Algorithm for the k Minimum Spanning Tree Problem in the Plane. In *Proc. 26th ACM Symp. on Theory of Computing* (STOC'94), Montreal, May 1994, pp 432–438.

[GJ79] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-completeness.* W. H. Freeman and Co., San Francisco, CA, 1979.

[Go85] T. F. Gonzalez, Clustering to Minimize the Maximum Intercluster Distance. *Theoretical Computer Science*, 38:293–306, 1985.

[HM79] G. Y. Handler and P. B. Mirchandani, *Location on Networks: Theory and Algorithms.* MIT Press, Cambridge, MA, 1979.

[HS86] D. S. Hochbaum and D. B. Shmoys, A Unified Approach to Approximation Algorithms for Bottleneck Problems. *Journal of the ACM*, 33(3):533–550, July 1986.

[KN+95a] S. O. Krumke, H. Noltemeier, S. S. Ravi and M. V. Marathe, Compact Location Problems with Budget and Communication Constraints. In *1st International Conference on Computing and Combinatorics,* X'ian, China, LNCS Volume 959, Springer Verlag, August, 1995, Pages 510-519.

[KN+95b] S. O. Krumke, H. Noltemeier, S. S. Ravi and M. V. Marathe, Bicriteria Problems in Location Theory. In *21st Workshop in Graph Theoretic Concepts in Computer Science (WG'95)*, Aachen, Germany, June 1995.

[KN+95c] S. O. Krumke, H. Noltemeier, S. S. Ravi and M. V. Marathe, Bicriteria Compact Location Problems for Trees (in preparation), Nov. 1995.

[KP93] G. Kortsarz and D. Peleg, On choosing a dense subgraph In *Proceedings of the 34th Annual IEEE Symposium on the Foundations of Computer Science (FOCS'93)*, 1993, pages 692–703.

[Lee82] D. T. Lee, On k-nearest neighbor Voronoi diagrams in the plane. *IEEE Trans. Comput.*, C–31:478–487, 1982.

[LV92] J. H. Lin and J. S. Vitter, ε–Approximations with Minimum Packing Constraint Violation. In *ACM Symposium on Theory of Computing (STOC'92)*, May 1992, pp 771–781.

[Meg83] N. Megiddo, *Applying parallel computation algorithms in the design of serial algorithms*, Journal of the ACM **30** (1983), no. 4, 852–865.

[MR+95] M. V. Marathe, R. Ravi, R. Sundaram, S. S. Ravi, D. J. Rosenkrantz and H. B. Hunt III, Bicriteria Network Design Problems. In *Proceedings of the 22nd International Colloquium on Automata Languages and Programming (ICALP'95)*, LNCS 944, Springer Verlag, July 1995, pp 487–498.

[MF90] P. B. Mirchandani and R. L. Francis, *Discrete Location Theory.* Wiley–Interscience, New York, NY, 1990.

[PS85] F. P. Preparata and M. I. Shamos, *Computational Geometry: An Introduction.* Springer–Verlag Inc., New York, NY, 1985.

[RKM+93] V. Radhakrishnan, S. O. Krumke, M. V. Marathe, D. J. Rosenkrantz and S. S. Ravi, Compact Location Problems. In *Proceedings of the 13th Conference on the Foundations of Software Technology and Theoretical Computer Science (FST&TCS'93)*, Volume 761 of *LNCS*, pages 238–247, December 1993.

[RMR+93] R. Ravi, M. V. Marathe, S. S. Ravi, D. J. Rosenkrantz and H. B. Hunt III, Many birds with one stone: Multi-objective approximation algorithms. In *Proceedings of the 25th Annual ACM Symposium on the Theory of Computing (STOC'93)*, May 1993, pages 438–447.

[RRT94] S. S. Ravi, D. J. Rosenkrantz and G. K. Tayi, Heuristic and Special Case Algorithms for Dispersion Problems. *Operations Research*, Vol. 42, No. 2, March-April 1994, pp 299–310. (An extended abstract of this paper appeared in *Proceedings of the 1991 Workshop on Algorithms and Data Structures* (WADS'91), Ottawa, Lecture Notes in CS, Vol. 519, Springer-Verlag, New York, NY, August 1991, pp 355–366.)

[RRT95] D. J. Rosenkrantz, S. S. Ravi and G. K. Tayi, Capacitated Versions of Dispersion Problems (in preparation), Nov. 1995.

[RR+94] R. Ravi, R. Sundaram, M. V. Marathe, D. J. Rosenkrantz and S. S. Ravi, Spanning trees short or small. In *Proceedings, Fifth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'94)*, Jan. 1994, pp 546–555. (Complete version to appear in *SIAM Journal on Discrete Mathematics.*)

[RSL77] D. J. Rosenkrantz, R. E. Stearns and P. M. Lewis II, An Analysis of Several Heuristics for the Traveling Salesman Problem. *SIAM J. Computing*, Vol. 6, No. 3, 1977, pp 563–581.

[ST93] D. B. Shmoys and E. Tardos, Scheduling unrelated parallel machines with costs. In *Proceedings of the 4th Annual ACM–SIAM Symposium on Discrete Algorithms (SODA'93)*, Jan. 1993, pp. 438–447.

[Ta91] A. Tamir, Obnoxious Facility Location on Graphs. *SIAM J. Discrete Mathematics*, Vol. 4, 1991, pp 550–567.

[ZL93] A. A. Zelikowsky and D. D. Lozevanu. Minimal and bounded trees. *Acad. Romano-Americane*, Kishinev, 1993, pp. 25–26